BasiCode

MIT PROGRAMMEN AUF SCHALLPLATTE FÜR HEIMCOMPUTER

```
1 O. "D:*.*",6,0
2 FOR I=1 TO 60
3 D$=R.(0)
4 PRINTD$
5 NEXT I
```

Federführender Autor:
Prof. Dr. rer. nat. Horst Völz

BASICODE

mit Programmen auf Schallplatte für Heimcomputer

AC 1
Z 1013
KC 85/1 und KC 87
KC 85/3 und KC 85/4
CPC 464 und CPC 6128
C 64 und C 128
C plus 4 und C 16
ATARI 800 XL/XE
ZX-Spectrum
IBM-kompatible PC's

Federführender Autor:
Prof. H. Völz (1. - 4.5., 5.11. - 6.6., 9. - Ende)

Autoren:
Dr. J. Baumann (7.2.)
A. Böhme (5.4.)
M. Duchrow (5.6.)
A. Graf (5.8.)
J. Haubrich (7.1.)
F. Heyder (5.7.)
M. van der Meer (5.9. und 5.10.)
B. Nickel (5.7.)
P. Reichelt (5.5.)
A. Zierott (5.1. - 5.3., 6.7.)
U. Zierott (5.1. - 5.3., 6.7.)

Basicode mit Programmen auf Schallplatte für Heimcomputer : AC 1, Z 1013, KC 85/1 u. KC 87, KC 85/3 u. KC 85/4, CPC 464 u. CPC 6128, C 64 u. C 128, Atari 800 XL/XE, ZX-Spectrum / [federführender Autor: H. Völz]. - Berlin : Verl. Technik, 1990. - 208 S. & 1 Datenplatte
ISBN 3-341-00895-0

ISBN 3-341-00895-0

1. Auflage
© Verlag Technik, Berlin, 1990
Lizenz 201 · 370/197/90
Printed in the German Democratic Republic
Satz: Verlag Technik, Berlin
Druck und buchbinderische Verarbeitung:
Druckzentrum Berlin, Grafischer Großbetrieb
Einbandgestaltung: *Gabriele Schwesinger*
LSV 3053
Bestellnummer: 554 342 0

BASICODE ist das Gemeinschaftswerk vieler Computerfreunde. Stellvertretend für alle gilt unser Dank besonders:

- der Stiftung BASICODE in den Niederlanden
ganz besonders Herrn Jacob Haubrich und Klaas Robers für die Bereitstellung aller Unterlagen und die fachliche Beratung.

- dem Verlag Kluwer technische boeken
für die Erlaubnis, das Prinzip von BASICODE-3 in einem Buch zu veröffentlichen.

- dem Ministerium für Kultur
für die Förderung des Gesamtprojektes und Unterstützung des Autorenkollektivs.

- dem Sender Radio DDR
für die populäre Verbreitung des BASICODE-Gedankens in der DDR.

- dem Verlag Technik
für die Beratung und Organisation bei der Herstellung dieses Buches.

- dem VEB Deutsche Schallplatten
für die Herstellung der Daten-Schallplatte mit den Computerprogrammen, wobei neue Aufzeichnungstechnologien angewandt wurden.

- dem gesamten Autorenkollektiv und den vielen Computerfreunden, die unter Leitung von Prof. Völz schöpferisch zusammengearbeitet haben.

Wir hoffen, mit diesem Buch vielen Computerfreunden Voraussetzung und Anregung zur schöpferischen und freundschaftlichen Zusammenarbeit zu geben.

Das Autorenkollektiv

Inhaltsverzeichnis

Vorwort 9

1. Was ist BASICODE ? 11

2. Grundlagen von BASICODE 13
2.1. Der Umgang mit BASICODE 14
2.2. Erste Befehle im Text-Mode 18
2.3. Erweiterter TEXT-Mode 21
2.4. Der Grafik-Mode 23
2.5. Die Tonausgabe 28
2.6. Umgang mit Daten-Files 29
2.7. Bemerkungen zum Bascoder 31
2.8. Ergänzungen zu den GOSUB-Routinen 31
2.9. Hinweise zu den Variablen 34

3. Hinweise zur optimalen Strukturierung in BASICODE 35
3.1. Einige Grundregeln der strukturierten Programmierung 36
3.2. Grundstrukturen für Programme 38
3.3. Hinweise zur Fehlersuche bei BASICODE 44
3.4. Bemerkungen zur garbage collection 45
3.5. Optimierung von Programmen 46

4. Das Kassetteninterface von BASICODE 49
4.1. Bit-Niveau 50
4.2. Byte-Niveau 53
4.3. Struktur von ASCII-Files und Programmen 54
4.4. Struktur von Datenfiles 55
4.5. Umgang mit Kassetten und Aufzeichnungen 56
4.6. Umgang mit Kassettenrecordern 59

5. Beschreibung der Bascoder 61
5.1. Bascoder für den KC85/3 61
5.2. Bascoder für den KC85/4 73
5.3. Bascoder für den KC87 83
5.4. Bascoder für den Commodore C 64 92
5.5. Bascoder für Commodore C plus/4 und C-16 (64k) 96
5.6. Bascoder für den Z1013 98
5.7. Bascoder für den Amateurcomputer AC 1 112
5.8. Bascoder für ATARI 800XL/XE 600XL 130XE 121
5.9. Bascoder für den Sinclair ZX-Spectrum 129
5.10. Bascoder für die Schneider-CPC-Rechner 133
5.11. Bascoder für 16-Bit-Rechner 136

6. Beispiel-Programme 141
6.1. Römische Zahlen 141
6.2. Humorvolle Zeilen 145
6.3. Gedichte in freier Prosa 147
6.4. Ein Miniatlas 152
6.5. Der Globus-Teil 158
6.6. Hiddenline-Darstellung 161
6.7. MOZART - Ein musikalisches Würfelspiel 166

7. Die Geschichte 173
7.1. Die Entwicklung von BASICODE 173
7.2. Computersendungen im Rundfunk der DDR 175

8. Die Rechte der BASICODE-Autoren 181

9. Vorstellung der Autoren 183
9.1. Holländische Autoren 183
9.2. DDR-Autoren 185

10. Hinweise zur Nutzung der Schallplatte 191

11. Tabellen 193
11.1. In BASICODE gültige BASIC-Befehle 193
11.2. Abweichungen bei den erlaubten BASIC-Befehlen 193
11.3. Kurze Übersicht der BASICODE-GOSUB-Routinen 195
11.4. Nicht erlaubte BASIC-Befehle der KC-Rechner 196
11.5. Zeilen-Verwendung beim BASICODE 196
11.6. In BASICODE verbotene Variablen 197
11.7. Im Bascoder verwendete Variablen mit besonderer Bedeutung 197
11.8. Kleines Fachwörterbuch 197
11.9. Daten verschiedener Rechnertypen 203

Literaturverzeichnis 204

Sachwörterverzeichnis 205

Vorwort

Die Anzahl der in der DDR "privat" verfügbaren Rechner wird zur Zeit auf ca. zweihunderttausend in weit mehr als zehn verschiedenen Typen geschätzt. Ähnlich groß dürfte auch die Anzahl der zusätzlichen Kleinstrechner in Schulen, Ausbildungsstätten, Computer-Kabinetten, -Arbeitsgemeinschaften usw. sein. Es kann also damit gerechnet werden, daß in unserer Republik rund eine Million Bürger sich in ihrer Freizeit mit Rechnern beschäftigen. Der Austausch von Gedanken, Programmen und Daten ist hierbei jedoch zum Teil sehr erschwert. Jeder Rechnertyp arbeitet mehr oder weniger abweichend. Diese Probleme lassen sich mit BASICODE in eleganter Weise beheben. Hinzu kommt, daß auch BASICODE in vorzüglicher Weise die Software-Verbreitung über den Rundfunk ermöglicht. Gerade für Anfänger, Arbeitsgemeinschaften und Schulen stellt folglich BASICODE eine nahezu ideale Lösung dar. In Kurzform berichtet hierüber Kapitel 1, während Kapitel 2 auf Details der Programmierung eingeht und Kapitel 4 sich dem universellen Daten- und Programmaustausch widmet.

BASICODE wurde ab 1979 von holländischen Spezialisten entwickelt, dabei erfolgte ständig eine Weiterentwicklung entsprechend dem Stand der Technik. Heute wird BASICODE-3 verwendet und an Weiterentwicklungen wird gearbeitet. Die fortschreitende Entwicklung wird aber immer auf dem Vorangegangenen aufgebaut. Mit BASICODE ist daher auch der Weg in die Zukunft gesichert. Dies wird noch deutlicher mit der in Kapitel 7. dargestellten Geschichte von BASICODE.

Seit September 1989 verwendet der Rundfunk der DDR bevorzugt BASICODE. Er hat sich inzwischen auch hier, ähnlich wie in den Niederlanden, vorzüglich bewährt. Das Hörerecho ist unwahrscheinlich groß. Bisher gab es aber nur ein kurzes Material des Rundfunks. Das hier vorliegende Buch soll die inzwischen deutlich gewordene Lücke bezüglich mehr Details schließen. Die besonderen Vorteile von BASICODE dürften so noch mehr wirksam werden. Sie können etwa in den folgenden drei Punkten zusammengefaßt werden.

- Die Kommunikationsbarriere zwischen verschiedenen Rechnertypen wird behoben.
- Der Rundfunk erreicht mit seinen Sendungen alle Rechnernutzer.
- Programme brauchen für alle Rechnertypen nur einmal entwickelt zu werden.

Nun benutzt BASICODE neben seinen vielen anderen Spezifika (siehe Kapitel 1.) zur Programmierung eine BASIC-ähnliche Sprache: ca. 50 BASIC-Befehle sind direkt übernommen und weitere 30 werden auf spezifische Weise durch Subroutinen realisiert. Dadurch können sich aber auch von BASIC her bekannte Schwächen auswirken. Deshalb wurde hier - genau wie bei den Rundfunk-Lehrgängen - großer Wert auf einen guten Programmierstil gelegt. Mit dem Kapitel 3. wird ein Versuch unternommen, die strukturierte Programmierung weitgehend unter dem Gesichtspunkt von BASICODE einzuführen.

Für BASICODE existiert in den Niederlanden eine gewinnfrei arbeitende Stiftung unter dem Vorsitz von Klaąs Robers, der auch entscheidender Initiator des BASICODE ist. Von dieser Stiftung hat der Rundfunk der DDR die Rechte zur Anwendung erworben und bei der Einführung eine umfangreiche selbstlose Hilfe erfahren.

Für BASICODE wird eine für jeden Rechnertyp spezifische Software, der Bascoder, benötigt. Die neuesten Varianten sind auf der anliegenden Schallplatte vorhanden. Wie sie genutzt wird, weist das Kapitel 11. aus. Diese Software wurde für viele Rechnertypen von niederländischen und DDR-Experten entwickelt. Einige rechtliche Fragen zum Umgang mit dieser Software behandelt Kapitel 8.

Dieses Buch entstand durch Zusammenarbeit in einem breiten Kollektiv. Nur so konnten u.a. die unterschiedlichen Rechnertypen fachmännisch behandelt werden. Es wurde versucht, die Leistungen der einzelnen Autoren möglichst exakt auszuweisen. Ein individueller Dank ist infolge ihrer Vielzahl kaum möglich. Mein Dank gilt allen. Es herrschte im wesentlichen eine angenehme Arbeitsatmosphäre, die sicher mit diesem Buch nicht beendet sein wird. Das BASICODE-Kollektiv wird - wie es in Holland schon länger der Fall ist - ständig wachsen. Und auch der BASICODE wird sich weiterentwickeln, und so dürften bald neue Bücher erforderlich werden. Besonders möchte ich aber die Entwicklungsarbeiten an den Bascodern hervorheben. Mit den Entwicklern der Bascoder für alle KC-Rechner durch Andreas und Uwe Zierott sowie für den des Atari durch Andreas Graf hat sich ein inniger und besonders konstruktiver Kontakt herausgebildet. Die Unterstützung durch den Rundfunk der DDR, insbesondere von Dr. Joachim Baumann, hat für die schnelle Verbreitung von BASICODE gesorgt, und so konnten erkannte Fehler bereits eliminiert werden. Besonderer Dank gebührt auch den Herren der BASICODE-Stiftung, vor allem Herrn Jacob Haubrich und Klaas Robers. Sie haben jeden von uns mehrfach, intensiv und selbstlos beraten, alles nur erdenkliche Material bis zu Quellcodes, Testprogramme usw. zur Verfügung gestellt und auch die meisten Teile des Manuskripts mit großer Sorgfalt und vielen nützlichen Hinweisen durchgesehen.

Noch einige Bemerkungen zur Gestaltung des Buches:

Es wurde versucht, weitgehend eine einheitliche Linie zu erreichen. Bewußt wurde die Individualität der einzelnen Autoren berücksichtigt. Da dies in erster Linie nur die Beschreibung des Bascoders betrifft, dürfte es nicht störend wirken.

Damit die Anwender der unterschiedlichen Rechnertypen nur die Beschreibung ihrer Bascoder zu lesen brauchen, wurden hier Dopplungen in Kauf genommen.

Für Hinweise auf Mängel und Fehler ist das Autorenkollektiv dankbar.

H. Völz

1. Was ist BASICODE ?

Wer mit Rechnern umgeht, stößt sehr schnell auf das Problem der **Inkompatibilität**, d. h., die verschiedenen Rechnertypen besitzen recht unterschiedliche Eigenschaften. Ein Programm oder ein Datensatz des KC85/3 ist z. B. nicht auf den PC1715 übertragbar. Diese Problematik beginnt bereits bei den Datenträgern. Der KC-Rechner verwendet Tonbandkassetten zur Speicherung, der PC1715 dagegen Disketten. Aber auch schon bei den Kassetten sind die Datenaufzeichnungen so unterschiedlich, daß z. B. eine Kassette vom KC85/3 weder mittels eines ZX-Spectrum, Commodore noch Atari zu lesen ist. Ganz ähnliches gilt auch bei den Disketten.

Genau für diese Probleme ist BASICODE die Lösung. Seine Entwickler schufen zunächst ein spezielles Datenformat für die Kassettenaufzeichnung und Rundfunkübertragung. Es wurde so gestaltet, daß es möglichst einfach für alle Rechner durch ein Zusatzprogramm realisiert werden kann. Dieses Zusatzprogramm heißt Bascoder. So kann dann das neue Format von allen Rechnern geschrieben und gelesen werden. Es wurde aber auch so gestaltet, daß es vorteilhafte Eigenschaften für die Rundfunkübertragung besitzt.

Die **Kassette** ist bei dieser Methode keineswegs ein veraltetes Medium. Einmal ist sie nämlich äußerst preiswert. Denn es ist viel leichter, an einem Rechner einen Kassettenanschluß zu realisieren und einen Kassettenrecorder zu verwenden als ein Diskettenlaufwerk und dessen Steuerung. Schließlich läßt sich auch eine Rundfunksendung unmittelbar auf Kassette aufzeichnen. Bei einer Diskette wären hierzu komplizierte und spezielle Zusatzeinrichtungen notwendig. Nur in sehr seltenen Fällen ist für den Kassettenanschluß auch etwas zu bauen. Zum Beispiel besteht die entsprechende Hardware für viele Rechner, die keinen Kassettenanschluß mehr besitzen, nur aus wenigen Widerständen und Kondensatoren.

Mit dem einheitlichen **Datenformat** können nun zwar leicht Daten, aber nur selten Programme ausgetauscht werden. Bei jeder heute existierenden Sprache gibt es nämlich mehrere nicht voll kompatible (untereinander verträgliche) Dialekte. Das gilt besonders für die bei Heimcomputern verwendete Sprache BASIC. Deshalb schufen die Entwickler von BASICODE auch eine genau definierte **Sprache**, die von jedem Rechner mittels des oben genannten Bascoders verstanden wird.

Doch auch mit dem einheitlichen Datenformat und der genormten Sprache ist noch nicht die volle Austauschbarkeit von Programmen möglich. Die verschiedenen Rechner besitzen nämlich noch unterschiedliche Bildschirme und manchmal auch Tastaturen. Beim KC85/3 existieren z. B. 32 Zeilen mit 40 Zeichen je Zeile. Beim PC1715 sind 80 Zeichen je Zeile möglich, der KC87 benutzt nur 24 Zeilen usw. Noch komplizierter wird es, wenn auf dem Bildschirm Grafik realisiert werden soll. Hier ist jedoch nicht nur die Auflösung, d. h. die Anzahl der Pixel (mögliche Bildpunkte), unterschiedlich. Der KC87 läßt z. B. keinen direkten Zugriff zu den Bildpunkten zu. Er verfügt nur über eine mittelbare (Pseudo-) Grafik. Der KC85/3 ermöglicht, 320 * 256 = 81920 Bildpunkte direkt zu benutzen. Hochleistungsrechner ermöglichen manchmal sogar eine Million. Es ist selbst für den Fachmann erstaunlich, daß auch hier die Entwickler von BASICODE eine einheitliche Lösung fanden. Schließlich wurde so etwas auch noch für den Ton gefunden. Ein Bildprogramm erzeugt daher mittels BASICODE auf allen Rechnern ein **einheitliches Bild**, ein Musikprogramm eine **gleichklingende Musik.**

Natürlich besitzt BASICODE auch **Nachteile und Grenzen.** Sie sind jedoch vergleichsweise gering und betreffen folgende Punkte:

Erstens ist für jeden Rechner ein zusätzliches spezifisches Programm - der "**Bascoder**" - notwendig. Er steht für die wichtigsten Rechnertypen (egal welcher Herkunft) zur Verfügung und wird für weitere entwickelt. Mit diesen Bascodern entstehen aber auch Vorteile. Sie sind meist in hocheffizientem Maschinencode geschrieben, und so ist BASICODE zuweilen erheblich schneller als BASIC.

Zweitens ist das Programmieren in BASICODE etwas **ungewohnt**. Es müssen nämlich die Vorschriften streng eingehalten werden. Dabei kann vielfach der Rechner nicht alle "Fehler" feststellen. Es können nämlich unter BASICODE einige Befehlsstrukturen auf einem Rechner fehlerfrei laufen, die andere Rechner nicht akzeptieren. Die diesbezüglichen Probleme sind in diesem Buch genau besprochen. Weiter existieren im Rundfunk der DDR sehr umfangreiche Testprogramme auf unerlaubte Befehle.

Drittens benötigen wenige Rechner für den Kassettenanschluß eine geringfügige zusätzliche **Hardware**.

Viertens besitzt BASICODE keinen **Compiler** zur Erzeugung kommerzieller Programme. Dennoch können unter BASICODE - wie auch schon die Beispielprogramme zeigen - sehr anspruchsvolle Lösungen realisiert werden.

Kurz **zusammengefaßt** ist BASICODE also keine neue Sprache, sondern **ein ganzes System von Festlegungen**, das in raffinierter Weise auf allen Rechnern die gleiche Wirkung hervorruft und so die Kommunikation zwischen allen Rechnern ermöglicht. Um dies zu erreichen, muß für jeden Rechner ein spezielles Programm, der Bascoder existieren. Er muß zunächst in den Rechner geladen werden. Dann ist er für die Norm BASICODE bereit und kann BASICODE-Programme und Daten lesen und schreiben, erzeugen und abarbeiten. Dies alles wurde ab 1979 von holländischen Spezialisten entwickelt und steht uns jetzt zur Verfügung. BASICODE ist aber auch kein starres System. Es entwickelt sich ständig mit dem Stand der Technik weiter. So existiert heute für uns BASICODE3. Jedoch werden Weiterentwicklungen schon vorbereitet, und sie werden dann weitere einheitliche Leistungen vollbringen. In jedem Fall wird diese Entwicklung aber immer so gestaltet, daß Altes dadurch nicht seine Bedeutung verliert. Es kommen nur neue Möglichkeiten hinzu. Dies ist ein Grund, daß BASICODE auch in der Zukunft weiter besteht. Es lohnt sich also, BASICODE zu nutzen.

2. Grundlagen von BASICODE

In diesem Kapitel wird der Sprachteil von BASICODE behandelt. Er schafft also die notwendigen Grundlagen für das Programmieren unter BASICODE.

Der Sprachteil von BASICODE ist in vielen Punkten BASIC ähnlich und besitzt damit auch dessen Vor- und Nachteile. Die Vorteile betreffen u. a.:

- die leichte Erlernbarkeit
- die hohe Interaktivität
- die effiziente Fehlerbehandlung.

Die Nachteile betreffen im wesentlichen die Möglichkeit, einen schlechten Programmierstil zu benutzen und das Fehlen eines Compilers. Bezüglich des Programmierstils existiert in diesem Buch das Kapitel 3. Das Fehlen eines Compilers begrenzt die Möglichkeiten zur Schaffung kommerzieller Programme. Dennoch sind in BASICODE durchaus anspruchsvolle Programme zu schreiben. In erster Linie ist aber BASICODE ohnehin für Hobby und Ausbildung entwickelt worden.

Bei diesem Kapitel wird **vorausgesetzt**, daß der Leser über die Grundkenntnisse des Programmierens in BASIC verfügt. Falls dies nicht der Fall ist, wäre es nützlich, sich z. B. über Bücher wie [2, 8, 9, 10] oder in Arbeitsgemeinschaften gleich zu Beginn der Beschäftigung mit BASICODE Kenntnisse in BASIC anzueignen. Die Anfangserfahrungen sind dabei gleichermaßen gut in BASICODE wie in BASIC zu gewinnen.

Der **Sprachteil** von BASICODE besteht nämlich aus zwei unterschiedlichen Komponenten:

Reichlich 50 Befehle sind in BASICODE weitgehend mit den entsprechenden BASIC-Befehlen identisch. Sie realisieren so etwas wie ein Minimal-BASIC, nämlich jene Menge, die in allen BASIC-Dialekten einheitlich gegeben ist. Diese Befehle sind in Tab. 11.1 zusammengestellt, und Tab. 11.2 faßt mögliche Abweichungen hierzu zusammen. Dies ist aber auch ziemlich genau jene Menge, mit der man BASIC erlernt.

Der zweite Teil besteht aus etwa 30 Befehlen, welche indirekt über GOSUB-Routinen realisiert werden. Sie betreffen bezüglich BASIC drei Befehlsgruppen:

- BASIC-Befehle, die sich auf mehreren Rechnertypen unterschiedlich verhalten
- BASIC-Befehle, die nur in einigen BASIC-Dialekten existieren
- Spezifika von BASICODE, die in keinem BASIC existieren.

Durch diese GOSUB-Routinen erlangt BASICODE eine beachtlich hohe Leistungsfähigkeit. Da die Standard-BASIC-Befehle auch anderweitig leicht zu erlernen sind, wird mit diesem Kapitel versucht, systematisch die Spezifika der GOSUB-Routinen einzuführen. Eine Zusammenstellung aller in BASICODE-3 verfügbaren GOSUB-Routinen enthält Tab. 11.3.

Infolge der unterschiedlichen BASIC-Dialekte und auch weil nicht jeder Bascoder alle in BASICODE unzulässigen BASIC-Befehle als Fehler ausweist, ist es notwendig, einige BASIC-Befehle konkret in BASICODE-Programmen zu verbieten. Sie sind in Tab. 11.4 zusammengefaßt. In BASICODE werden weiter einige Variablen ganz spezifisch verwendet. Dies sollte bei der Verwendung von Variablen beachtet werden. Die entsprechenden Fakten enthalten die Tab. 11.6 und 11.7.

2.1. Der Umgang mit BASICODE

Für jeden Rechnertyp existiert ein Bascoder, der von einem entsprechenden Experten entwickelt wurde. Dabei sind natürlich auch etwas unterschiedliche Handhabungen entstanden. Sie sind teilweise durch die Hardware des jeweiligen Rechners bedingt. Stärker sind jedoch Ambitionen des Entwicklers eingegangen. Aus diesen Gründen läßt sich keine voll allgemeingültige Anleitung für den Umgang mit BASICODE geben. Im Kapitel 5. ist für jeden Rechnertyp sowohl eine leicht verständliche und schnell realisierbare als auch eine detaillierte Einführung für jeden einzelnen Rechnertyp vorhanden. **Ungeduldige** sollten also dort beginnen! Natürlich gibt es aber auch weitgehend allgemeingültige Grundsätze. Sie werden im folgenden dargestellt. Zuvor sei jedoch noch einiges zum normalen **BASIC-Betrieb** eines Rechners ausgesagt. Hierfür existieren nämlich drei Betriebsebenen:

A1. Das Betriebssystem.
Bei den KC-Rechnern CAOS bzw. OS. Hier können z. B. Maschinenprogramme geladen und abgearbeitet werden. Ein solches Programm ist z. B. auch der übliche BASIC-Interpreter.

A2. Der BASIC-Direkt-Mode.
Er wird meist über den Aufruf

```
>BASIC
```

erreicht und ermöglicht ähnlich wie ein Taschenrechner direkte Befehls-Eingaben wie LIST, PRINT 3+5 usw. Nach der Abarbeit erscheint OK oder bei anderen Rechnern READY.

A3. Der Programm-Mode.
Er benötigt zusätzlich ein geladenes BASIC-Programm und wird dann durch den Direkt-Befehl RUN eingeleitet.

Bei BASICODE existiert eine Ebene mehr. Sie wird durch ein Spezialprogramm, den Bascoder, bewirkt. Der Bascoder wird bei fast allen Rechnern in der A2-Ebene geladen. Hierbei wird bei einigen Rechner-Typen automatisch ein neuer Zustand (der BASICODE-Mode) erreicht. In den anderen Fällen wird dieser Zustand aber über die Eingabe von RUN (oder einen anderen Befehl) erreicht. Der Bascoder wird also in der A2-Betriebsebene wirksam gemacht. Wenn dann der BASICODE-Mode erreicht ist, bestehen wieder folgende Möglichkeiten:

- Es können die 50 BASIC-Befehle und die 30 Subroutinen unmittelbar genutzt werden (Direkteingabe, Taschenrechner-Betrieb).
- Es kann ein BASICODE-Programm geladen werden und anschließend wie in BASIC mit RUN gestartet werden.
- Es kann ein BASICODE-Programm durch Eingabe über die Tastatur erzeugt und erprobt werden.

Beim **BASICODE** existiert also im Vergleich zu BASIC noch eine Betriebsebene mehr:

B1. Betriebssystem
B2. BASIC-Direkt-Mode
B3. Der über den Bascoder erreichte BASICODE-Mode
B4. Die Nutzung des BASICODE-Mode für BASICODE-Programme

Der BASICODE-Mode wird also mittellbar über den Bascoder bewirkt und ist ein besonderer (neuartiger) Zustand Ihres Rechners. Hier besitzen Sie einen Rechner mit neuen Eigenschaften. In

diesem Mode gleicht Ihr Rechner allen anderen Rechnertypen, die sich dazu ebenfalls im BASICODE-Mode befinden. Der Fachmann sagt: Ihr Rechner ist jetzt eine neue "virtuelle Maschine". Der Bascoder verleiht ihm in diesem Zustand also jene Eigenschaften, durch die BASICODE gekennzeichnet ist.

Dieser besondere Zustand ist zur Beendigung des BASICODE-Mode folglich auch wieder zu verlassen. Aus diesem Grunde enden BASICODE-Programme auch nicht wie übliche BASIC-Programme, z. B. durch ein END bzw. STOP oder gar ohne besondere Hinweise, indem das Programm ganz einfach aufhört. Ein BASICODE-Programm muß mit GOTO 950 enden. Mit dieser Subroutine gelangt der Rechner vom BASICODE-B3-Zustand wieder in den BASIC-Direktmode.

Für das Arbeiten mit BASICODE ist es noch wichtig, über zwei unterschiedliche File-Typen Bescheid zu wissen:

- ASCII-File (ASCII = amerikanischer Standard zur Zeichendarstellung)
- Token-File oder gleichwertige Bezeichnung "interne" Darstellung.

Ein Programm möge die folgende Zeile enthalten:

```
1Ø3Ø FOR X=1 TO 2Ø
```

Sie existiert zunächst genau in dieser Weise als ASCII-Zeile. Jedes Zeichen ist genau durch eine HEX-Zahl dargestellt: A z. B. 41. Die Zeile würde also folgende HEX-Zahlen ergeben, mit denen sie zunächst bei der Eingabe wirklich im Speicher abgelegt wird:

```
31 3Ø 33 3Ø 2Ø 46 4F 52 2Ø 58 3D 31 2Ø 54 4F 2Ø 32 3Ø
```

Auch die Leerzeichen (Space) sind mit dem Wert 20 hierin enthalten. Diese Darstellung ist für den BASIC-Interpreter ungünstig. Befehle wie FOR, TO usw. sind schwer erkennbar. Sie werden daher meist durch ein Zeichen (Token) ersetzt, das größer als 7F ist. Außerdem wird auch meist die Zeilennummer in eine Binärzahl verwandelt. Für die KC-Rechner entsteht so die interne bzw. Token-Darstellung:

```
4Ø Ø6 81 2Ø 58 B4 31 2Ø A6 2Ø 32 3Ø
```

Diese Folge ist also wesentlich kürzer. 1030 = 31 30 33 30 ist durch 40 06, FOR = 46 4F 52 durch 81 usw. ersetzt.

BASICODE-Programme existieren zunächst immer in der ASCII-Form. So werden sie übertragen, gespeichert, geladen usw. Diese Methode garantiert zwei Vorteile: Alle ASCII-Programme dieser Art sind leicht zwischen den Rechnern aller Typen austauschbar. Wie noch im Kapitel 4. gezeigt werden wird, ergibt sich so eine hohe Störsicherheit.

Mit einem ASCII-Programm kann Ihr Rechner nun aber in BASICODE genauso wie in BASIC direkt nichts anfangen. Es muß eine Übersetzung in die interne Darstellung erfolgen. Erst dann kann er es abarbeiten. Deshalb erfolgt im BASICODE-B3-Zustand folgender Ablauf:

- Laden des ASCII-Programms
- Übersetzen des ASCII-Programms in die interne Darstellung
- Abarbeiten des Programms.

In der Regel werden Sie nichts von diesem Ablauf im Detail merken. Im Anschluß an das Laden erfolgt meist automatisch das Übersetzen und z. T. auch das Abarbeiten. Abweichungen kommen jedoch dann vor, wenn beim Laden Fehler auftreten. Dann wird meist die Übersetzung mit Fehlerangabe abgebrochen. Auch jetzt macht sich sofort wieder der Vorteil des ASCII-Files bemerkbar. Das ASCII-File ist unmittelbar lesbar, und meist werden Sie aus dem Zusammenhang den Fehler noch erkennen und damit korrigieren können. So sind fehlerhafte Übertragungen vom Rundfunk oder von der Kassette leicht fehlerfrei zu machen.

Andererseits folgt aus dieser Betrachtung aber auch, wie bei der Programmspeicherung vorzugehen ist. Es gibt hier wieder die beiden Möglichkeiten:

- ACII-File (BASICODE-Format)
- Token-File (interne Darstellung).

Die Speicherung als ASCII-File liefert also ein BASICODE-Programm, das auf allen Rechnern in BASICODE genutzt werden kann. Das Token-File läuft dagegen nur auf Ihrem Rechnertyp. Damit das längere ASCII-File nicht zu lang wird, werden für das BASICODE-Programm alle unnötigen Space (oben 20) entfernt. Notwendig sind Leerzeichen nur vor GOSUB, GOTO und THEN und das auch nur, wenn zuvor kein ":" oder ")" steht. So ist auch ein BASICODE-Programm recht kurz. Für eine bestimmte Zeit muß also das ASCII-File im Eingabe-Puffer des Rechners stehen. Dieser ist nun wieder für die einzelnen Rechner unterschiedlich groß, und bei BASICODE muß leider auch auf den schwächsten Rechner Rücksicht genommen werden. Deshalb darf die BASICODE-Zeile nicht mehr als **sechzig Zeichen** - einschließlich der Space, Zeilennummer usw. - enthalten. Auf weitere Besonderheiten wird noch im folgenden eingegangen.

Weitere Grundsätze von BASICODE seien nun mit dem ersten Programm "HYDRA" demonstriert:

Es beginnt mit der Zeile 1000. Dieses ist in jedem BASICODE-Programm die erste Zeile. Das GOTO 20 in dieser Zeile initialisiert Grundbedingungen für BASICODE. Hierdurch werden typische Parameter und Bedingungen für den BASICODE-Mode festgelegt und die für den Rechner notwendigen Spezifika ausgelöst, die Variablen werden gelöscht und weitere notwendige Aktionen bewirkt. Außerdem wird dabei eine "CLEAR"-Anweisung gemäß dem normalen BASIC ausgeführt. A = 100 bedeutet hierfür, daß ein Stringraum der Größe hundert Byte reserviert wird. Kleinere Werte für A sollten nie verwendet werden, größere sind je nach den Notwendigkeiten und dem verfügbaren Speicherbereich zu wählen.

Von GOTO 20 führt nach diesen Initialisierungen der Bascoder den Rücksprung zur Zeile 1010 aus. Hier beginnt schließlich die Abarbeitung des "eigentlichen" BASICODE-Programms. Bei diesem speziellen Programm gelangt man nach der Anzeige "Hydra-Problem" in die Zeile 1030. Für die Eingabe Z = 0 wird hier dieses BASICODE-Pogramm beendet, und es erfolgt daher mit GOTO 950 der Rücksprung in den Bascoder und die Überleitung des Rechners in den BASIC-Mode, normaler BASIC-Zustand. Es erscheint wieder OK bzw. READY.

GOTO 950 steht also für das Ende des Programms, und daher sind im Programm die BASIC-Befehle **RUN, END** und **STOP** verboten.

Verallgemeinert folgen aus diesen Ausführungen erste allgemeine Regeln für ein BASICODE-Programm.

Es muß immer je eine Zeile 1000 und 1010 existieren.

In der **Zeile 1000** sollte dabei der notwendige Stringraum über die Variable A festgelegt werden. Danach hat ein Sprung zur Zeile 20 zu erfolgen. Genau dieser Aufbau der Zeile 1000 ist notwendig. Sie soll aber auch noch ein REM und die Angabe des Namens des Programms enthalten.

Die **Zeile 1010** ist der eigentliche Beginn des BASICODE-Programms. Auch sie muß immer existieren, da der Bascoder nach der Initialisierung zu dieser Zeile springt.

Ein BASICODE-Programm endet immer mit **GOTO 950**. Wird ein BASICODE-Programm durch Fehler oder manuell unterbrochen, so ist die übliche Fortsetzung mit CONT meist möglich. Wenn sie nicht erfolgt, sollte durch direktes Eingeben von GOTO 950 wieder der BASIC-Zustand bewirkt werden.

BASICODE-Programme verwenden - wie oben bereits erwähnt - eine sehr dichte Speicherung. In ihnen werden durch den Bascoder fast alle **Leerzeichen** automatisch entfernt. Deshalb sind BASICODE-Programme anfangs etwas schwer zu lesen. Dies wird in diesem Buch bei den Programm-Dokumentationen bewußt unterdrückt. Zur besseren Lesbarkeit sind zusätzliche Space

(Leerzeichen) eingefügt worden. Deshalb sollten Sie sich nicht wundern, wenn Ihr Programm-Listing auf dem Bildschirm oder im Ausdruck weniger schön aussieht.

Weiter sei erwähnt, daß es bei den Programmen in diesem Kapitel vor allem darauf ankommt, die Wirkung der einzelnen GOSUB-Routinen deutlich und didaktisch hergeleitet zu zeigen. Die Programme sind nur für diesen Zweck entworfen und besitzen als eigenständige Programme sonst kaum Bedeutung. Dann wären sie nämlich wesentlich länger und damit unübersichtlicher geworden. Vollständige Programme enthält Kapitel 6.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### Hydra ###
1Ø1Ø PRINT "Hydra-Problem": PRINT
1Ø2Ø INPUT "Eingabe einer Zahl:  Ø fuer Ende";Z
1Ø3Ø IF Z=Ø THEN 95Ø: REM Ende
1Ø4Ø IF(Z<>INT(Z))OR(Z<2)THEN PRINT"Fehler": GOTO 1Ø2Ø
1Ø5Ø Y=Z/2: IF Y=INT(Y) THEN Z=Y: GOTO 1Ø7Ø
1Ø6Ø Z=Z+Z+Z+1
1Ø7Ø PRINT Z;: IF Z>1 THEN 1Ø4Ø
1Ø8Ø PRINT: GOTO 1Ø2Ø
1Ø9Ø REM ----Programm-Ende------
3ØØØØ REM Das Hydra-Problem stammt von McCarthy.
3ØØ1Ø REM Es wird vermutet, dass alle ganzzahligen
3ØØ2Ø REM Eingaben zu 1 und damit zum Ende fuehren.
3ØØ3Ø REM Ein Beweis steht bis heute aus.
3ØØ4Ø REM Infolge der endlichen Arithmetik kann
3ØØ5Ø REM ein Rechner-Programm nur fuer einige Zahlen
3ØØ6Ø REM diese Hypothese unterstuetzen.
3ØØ7Ø REM Beobachten Sie das Zu- und Abnehmen
3ØØ8Ø REM der Zahlenwerte und versuchen Sie,
3ØØ9Ø REM das Programm zu verstehen.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 9.5.89
32Ø2Ø REM XT-compatibler Rechner
```

Über die vielen neuen Details wurde bisher noch nicht die eigentliche Aufgabe des Programms besprochen. Sie besteht im Folgenden: Es wird eine natürliche Zahl eingegeben. Sie kann gerade oder ungerade sein. Wenn sie gerade ist, wird sie durch zwei geteilt. Ist sie ungerade, so wird mit drei multipliziert und anschließend wird noch eins addiert. Dies erfolgt in den Zeilen 1040 bis 1060. Mit der so erhaltenen Zahl wird genauso weiter verfahren. Dies ist fortlaufend zu wiederholen, bis 1 erreicht wird. Es ist bis heute nicht bewiesen, ob die 1 für alle denkbaren Zahlen erreicht wird. Probieren Sie es einmal für einige. Ändern sie dies Programm so ab, daß sie das fortlaufende Kleiner und Größerwerden der Zahlen beobachten können. Zusätzliche Bemerkungen stehen in den Kommentarzeilen. Die Verwendung der Zeilennummern ist für BASICODE genau festgelegt (Abschnitt 11.5.). Ab Zeile 30000 stehen Erklärungen zum Programm und ab Zeile 32000 formale Fakten, wie Autor, Datum, Versionsnummer, verwendeter Rechner usw.

Es ist weiter noch eine Bemerkung zu den Programmen dieses Kapitels notwendig. Sie dienen nur der Demonstration und sind deswegen auf das Wesentlichste beschränkt. Dadurch sind sie auch immer noch leicht zu ändern und zu verbessern.

2.2. Erste Befehle im Text-Mode

Die nächsten vier Programme betreffen den Text-Mode. Damit sollen weitere GOSUB-Routinen eingeführt werden. Der zweite Mode - der Grafik-Mode - wird im Abschnitt 2.4. behandelt.

```
1000 A=100: GOTO 20: REM ### DATEN ###
1010 PRINT "Bildschirmdaten"
1020 PRINT"Text: Spalten * Zeilen =";HO;"*";VE
1030 PRINT"Pixel: X * Y ="; HG; "*"; VG; " = "; HG * VG
1040 PRINT"Taste betaetigen": GOSUB 210: GOTO 950
1050 REM ----Programm-Ende------
30000 REM Anzeige der vorhandenen Bildschirmwerte.
30010 REM Es existieren zwei Modi:
30020 REM Zeilen/Spalten fuer Text
30030 REM Pixel fuer Grafik
32000 REM -----------------
32010 REM H. Voelz; 10.5.89
32020 REM XT-compatibler Rechner
```

Das zuvor stehende Programm "DATEN" demonstriert drei Fakten:

1. Anhalten des Programms, bis eine Taste betätigt wird (GOSUB 210)
2. Anzeige der spezifischen Parameter des jeweiligen Rechners (Zeilen 1020 und 1030)
3. den Umgang mit BASICODE-spezifischen Variablen (HO,VE,HG,VG).

Mit GOTO 20 werden automatisch vier spezielle Variablen mit Daten belegt, die für den angewendeten Rechner typisch sind:

HO	(*horizontal*)	enthält die Anzahl der möglichen Spalten -1,
VE	(*vertikal*)	enthält die Anzahl der möglichen Zeilen -1,
HG	(*horiz. grafisch*)	enthält die Anzahl der möglichen Pixel in x-Richtung,
VG	(*vertik. grafisch*)	enthält die Anzahl der möglichen Pixel in y-Richtung.

Diese Werte werden in den Zeilen 1020 und 1030 angezeigt, bei den Pixeln auch ihre gesamte Anzahl. Damit ist das Programm eigentlich bereits zu Ende. Jedoch, wenn darauf GOTO 950 unmittelbar folgen würde, ginge der Rechner sofort in den BASIC-Mode über und alles wäre vom Bildschirm verschwunden. Deshalb muß man die Anzeige so lange sichtbar halten, bis eine Taste betätigt wird. Im normalen BASIC geschieht dies oft mit: INPUT A oder INPUT"";A. Bei beiden erscheint rechnerspezifisch meist der Cursor und zuweilen auch noch ein Fragezeichen. BASICODE besitzt einen besseren Befehl mit **GOSUB 210**. Das Programm wird es Ihnen zeigen.

Das nächste Programm "ZUTEXT" erzeugt Zufallstext und verteilt ihn unregelmäßig über den Bildschirm. Hierzu verwenden viele BASIC-Dialekte den Befehl RND. Da er aber nicht in allen Rechnern existiert, wird statt dessen in BASICODE die Anweisung **GOSUB 260** benutzt. Mit ihr wird in die Variable **RV** (*random value*) ein Zufallswert gemäß $0 <= RV < 1$ abgelegt.
Ein "RANDOMIZE" ist ebenfalls bei der Initialisierung über GOTO 20 vorhanden. Dadurch ist die Pseudozufallsfolge in ihrem Start bei RUN unbestimmt.

Die zufällige Anordnung auf dem Bildschirm erfolgt mittels eines Befehls, der PRINT AT ähnelt und durch **GOSUB 110** realisiert wird. Die Zeilen- und Spaltenpositionen werden hierbei mittels der Variablen HO und VE übergeben. Dies sind genau jene Variablen, die nach GOTO 20 die Spalten- bzw. Zeilenzahl enthalten. Deshalb werden diese Werte in 1010 auf die Variablen HT und VT zur Normierung der Zufallszahlen übergeben. So kann erreicht werden, daß bei jedem Rechner der Bildschirm gleichmäßig beschrieben wird. Mit CHR$(32+RV*59) werden zufällig die ASCII-

Zeichen von 32 bis 90 erzeugt. In dem Programm fällt das **GOSUB 100** in Zeile 1080 auf. Es entspricht im wesentlichen dem Löschen des Bildschirmes und holt erneut die Bildschirmparameter auf die o. g. Variablen zurück. Es setzt auch den Text-Mode.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### ZUTEXT ###
1Ø1Ø HT=HO: VT=VE
1Ø2Ø FOR X=1 TO 2ØØ
1Ø3Ø   GOSUB 26Ø: HO=RV*HT: GOSUB 26Ø: VE=RV*VT
1Ø4Ø   GOSUB 11Ø: GOSUB 26Ø: PRINT CHR$(32+RV*59);
1Ø5Ø NEXT X
1Ø6Ø HO=Ø: VE=Ø: GOSUB 11Ø
1Ø7Ø INPUT "Ø=Ende 1=weiter"; Z
1Ø8Ø IF Z=1 THEN GOSUB 1ØØ: GOTO 1Ø2Ø
1Ø9Ø GOTO 95Ø
11ØØ REM ----Programm-Ende------
3ØØØØ REM Es werden Zufallszeichen an zufaellige
3ØØ1Ø REM Bildschirmpositionen gelegt. Beobachten
3ØØ2Ø REM Sie das entstehende Muster und
3ØØ3Ø REM eventuell sich ausbildende Texte.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 9.5.89
32Ø2Ø REM XT-compatibler Rechner
```

Eine nützliche Anwendung der hier demonstrierten Prinzipien zeigt Abschnitt 2.3. Im Kapitel 6. ist ein leistungsfähigeres Programm der Textverarbeitung behandelt.

Die Übergabe der Pixelzahlen sollte wie in Zeile 1010 immer auf die Variablen **VT** und **HT** (*von T wie Text*) erfolgen.

Das folgende Programm "TASTE" verwendet zwei weitere BASICODE-Befehle, von denen einer nur selten ein Äquivalent im normalen BASIC besitzt.

1. Mit **GOSUB 150** wird eine in **SR$** (*reservierter String*) vorher abgelegte Zeichenkette auffällig auf den Bildschirm gebracht. Bei den meisten Bascodern ist dies eine inverse Zeichendarstellung. Zur besseren Hervorhebung werden an SR$ dabei noch je rechts und links drei Leerzeichen angefügt. In dem Programm wird aus dem Wert von HO nach **GOSUB 100** so die Lage des auffälligen Textes gemäß HO = INT((HO-LEN(SR$)-6)/2) berechnet, daß er bei jedem Rechner in der Mitte auf dem Bildschirm steht.
2. Mit **GOSUB 250** wird ein kurzer Ton (BEEP) erzeugt, der als Hinweis für besondere Ereignisse verwendet werden kann, z. B. wenn ein länger rechnendes Programm zum Ende gekommen ist.

Hauptsächlich dient das Programm jedoch zur genaueren Demonstration der Eingaberoutine **GOSUB 210**. Sie übergibt nämlich den Tastencode an zwei Variablen: **IN** erhält ASCII-Werte in leicht modifizierter Art. So werden z. B. meist nur die Zahlenwerte für Großbuchstaben übergeben, auch wenn Kleinbuchstaben betätigt wurden. **IN$** enthält das Zeichen, also z. B. "A" bzw. "a" bei IN = 65. Beide IN sind von INPUT abgeleitet. Ferner werden folgende Steuerzeichen übergeben:

13 für ENTER, CR,	127 für DELETE
28 für Cursor nach links	29 für Cursor nach rechts
30 für Cursor nach unten	31 für Cursor nach oben

Es ist zu beachten, daß diese Übergaben sehr rechnerspezifisch sein können. Deshalb ist ein Experimentieren mit dem Programm sehr nützlich. Beachten Sie beim Schreiben von BASICODE-Programmen die o. g. Einschränkungen.

Es sei noch erwähnt, daß die mit **GOSUB 210** verwandte Routine **GOSUB 200** bezüglich der Variablen sich gleichwertig verhält. Hierbei wird jedoch nicht die Tastenbetätigung abgewartet. Wenn beim Aufruf keine Taste berührt wurde, betragen IN = 0 und IN$ = "".

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### TASTE ###
1Ø1Ø SR$="Tastentest": HO=INT((HO-LEN(SR$)-6)/2)
1Ø2Ø VE=Ø: GOSUB 11Ø: GOSUB 15Ø: PRINT
1Ø3Ø PRINT"Tasteneingabe; Ø fuer Ende": GOSUB 25Ø
1Ø4Ø GOSUB 21Ø: PRINT IN$; IN: IF IN$<>"Ø" THEN 1Ø4Ø
1Ø5Ø GOSUB 25Ø: GOTO 95Ø
1Ø6Ø REM ----Programm-Ende------
3ØØØØ REM Test der Tastatureingabe gemaess GOSUB 21Ø
3ØØ1Ø REM und Demonstration von GOSUB 15Ø, 25Ø
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 9.5.89
32Ø2Ø REM XT-compatibler Rechner
```

Zuweilen ist es notwendig, die Tastatur bezüglich Eingaben zu sperren, damit ein Programm ohne äußere Störung, z. B. auch in Schulen oder Arbeitsgemeinschaften, ablaufen kann. Im Gegensatz zu den meisten BASIC-Dialekten bietet auch hier BASICODE mit **GOSUB 280** eine Lösung an, die das folgende Programm demonstriert. Ausgewertet wird hierbei die Variable **FR** (*Freigabe*). Sie muß vor dem Aufruf belegt werden:

FR = 0 STOP/BRK-Taste ist wirksam
FR = 1 STOP/BRK-Taste ist unwirksam

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### STOP ###
1Ø1Ø SR$="Stoptest": HO=INT((HO-LEN(SR$)-6)/2)
1Ø2Ø VE=Ø: GOSUB 11Ø: GOSUB 15Ø: PRINT: FR=1
1Ø3Ø PRINT "Stop: FR =";FR: GOSUB 28Ø
1Ø4Ø PRINT "Tasteneingabe; Ø fuer aendern": GOSUB 25Ø
1Ø5Ø GOSUB 21Ø: PRINT IN$; IN: IF IN$<>"Ø" THEN 1Ø5Ø
1Ø6Ø INPUT"Ø: stop moeglich; 1: nicht; 2: Ende"; FR
1Ø7Ø GOSUB 25Ø: IF FR<2 THEN 1Ø3Ø
1Ø8Ø GOTO 95Ø
1Ø9Ø REM ----Programm-Ende------
3ØØØØ REM Test auf Moeglichkeit der Stoptaste
3ØØ1Ø REM also CTRL C, BRK, Stop oder aehnlich
3ØØ2Ø REM erfolgt mit FR und GOSUB 27Ø
3ØØ3Ø REM Wenn Sie das Programm mit BRK usw.
3ØØ4Ø REM unterbrechen, denken Sie bitte daran,
3ØØ5Ø REM mit GOTO 95Ø Ihren Rechner wieder
3ØØ6Ø REM in den Normalzustand zu versetzen.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 9.5.89
32Ø2Ø REM XT-compatibler Rechner
```

2.3. Erweiterter TEXT-Mode

In diesem Abschnitt werden die restlichen GOSUB-Routinen des Text-Mode behandelt. Das Programm "TEXTUM" liest insbesondere über **GOSUB 220** Zeichen, die auf dem Bildschirm stehen, in die Variable IN zurück. Es besteht eine gewisse Verwandtschaft mit dem Befehl VPEEK im KC-BASIC. Außerdem werden wieder die Besonderheiten der Routinen **GOSUB 200** bzw. **GOSUB 210** wirksam. In IN$ werden zwar Groß- und Kleinbuchstaben zurückgemeldet, jedoch in IN nur die Werte der Großbuchstaben. Für "A" und "a" gilt also IN = 65. Solche Routinen sind u. a. günstig, wenn man menügesteuert wie im CAOS-Mode des KC85/3 oder KC85/4 arbeiten will. Das Programm "TEXTUM" macht allerdings einen anderen Gebrauch von dieser Routine. Zunächst verwendet es den Kernteil vom Programm "ZUTEXT". Danach werden die Zufallszeichen wieder vom Bildschirm gelesen und dicht aneinandergefügt. Hier wurden - trotz der Probleme, die sich beim C64 infolge seiner unüblichen Zeichenkodierung ergeben - die Kleinbuchstaben verwendet.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### TEXTUM ###
1Ø1Ø HT=HO: VT=VE: RESTORE
1Ø2Ø FOR I=1 TO 4: READ A$: PRINT A$: NEXT I
1Ø3Ø INPUT"Taste druecken"; A: GOSUB 1ØØ
1Ø4Ø FOR X=1 TO 2ØØ
1Ø5Ø   GOSUB 26Ø: HO=RV*HT: GOSUB 26Ø: VE=RV*VT
1Ø6Ø   GOSUB 11Ø: GOSUB 26Ø: PRINT CHR$(32+RV*95);
1Ø7Ø NEXT X
1Ø8Ø HO=Ø: VE=Ø: GOSUB 11Ø
1Ø9Ø GOSUB 25Ø: PRINT "Text zusammenfassen": BH=Ø: BE=1
11ØØ FOR VE=1 TO VT
111Ø   FOR HO=Ø TO HT
112Ø      GOSUB 22Ø: IF IN=32 THEN 117Ø
113Ø      BO=HO: HO=BH: BV=VE: VE=BE: GOSUB 11Ø
114Ø      PRINT CHR$(IN);
115Ø      HO=BO: VE=BV: GOSUB 11Ø
116Ø      BH=BH+1: IF BH=HT THEN BH=Ø: BE=BE+1
117Ø    NEXT HO
118Ø NEXT VE
119Ø GOSUB 25Ø: HO=Ø: VE=BV+1
12ØØ GOSUB 11Ø: PRINT "Ich bin fertig";
121Ø GOSUB 21Ø: GOTO 95Ø
122Ø REM ----Programm-Ende------
25ØØØ DATA "Der Commodore verfuegt ueber einen"
25Ø1Ø DATA "vom ASCII-Code abweichenden Zeichen-"
25Ø2Ø DATA "satz. Daher funktioniert dieses "
25Ø3Ø DATA "Programm bei ihm abweichend !"
3ØØØØ REM ----------------------
3ØØ1Ø REM Es werden Zufallsbuchstaben an zufaellige
3ØØ2Ø REM Bildschirmpositionen gelegt. Dann werden
3ØØ3Ø REM sie vom Bildschirm zurueckgelesen und
3ØØ4Ø REM hintereinander ausgegeben. Dabei werden
3ØØ5Ø REM Klein- in Grossbuchstaben gewandelt.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 9.5.89
32Ø2Ø REM XT-compatibler Rechner
```

Im normalen BASIC hat man bei Sortierprogrammen ein Problem mit den Klein- und Großbuchstaben. Durch ihren ASCII-Code werden sie nicht gleichwertig geordnet. BASICODE stellt u. a. zu diesem Zweck eine Routine **GOSUB 330** bereit. Sie verwandelt alle Kleinbuchstaben - die in SR$ enthalten sind - in Großbuchstaben. Dann ist ein exaktes Sortieren möglich. Genau dies nutzt das Programm "BUBBLE" aus.

```
1000 A=5000: GOTO 20: REM ### BUBBLE ###
1010 SR$="Einfacher Bubble-Sort": VE=0: M=21
1020 HO=INT((HO-LEN(SR$)-6)/2): GOSUB 110: GOSUB 150: PRINT
1030 PRINT: PRINT"bitte etwas warten"
1040 DIM A$(M), B$(M), A(M)
1050 FOR I=0 TO M
1060   GOSUB 260: N=INT(10*RV)+3: SR$=""
1070     FOR J=0 TO N
1080       GOSUB 260: IF RV<.5 THEN X=ASC("A")+RV*52
1090       IF RV>=.5 THEN X=ASC("a")+(RV-.5)*52
1100       SR$=SR$+CHR$(X)
1110     NEXT J
1120   A(I)=I: A$(I)=SR$: GOSUB 330: B$(I)=SR$
1130 NEXT I
1140 GOSUB 250: PRINT "Dies sind die Woerter:"
1150 FOR I=0 TO M: PRINT A$(I),B$(I): NEXT I
1160 PRINT "Nun wird sortiert :";
1170 FOR I=0 TO M: F=0
1180   FOR J=0 TO M-1-I
1190     IF B$(A(J))<=B$(A(J+1)) THEN GOTO 1210
1200     B=A(J): A(J)=A(J+1): A(J+1)=B: F=1
1210   NEXT J
1220  PRINT I;: IF F=0 THEN I=M
1230 NEXT I
1240 PRINT: GOSUB 250: PRINT "sortierte Woerter"
1250 FOR I=0 TO M: PRINT A$(A(I)): NEXT I
1260 GOSUB 210: GOTO 950
30000 REM ----------------------------------------
30010 REM Es wird der indirekte Bubble-Sort verwendet.
30020 REM Zur Vermeidung von garbage collection wird
30030 REM der Vergleich und Austausch ueber das Feld
30040 REM A(I) realisiert. Weiter werden die Woerter
30050 REM mit Grossbuchstaben in B$(I) abgelegt.
30060 REM Nach der Erzeugung werden beide Worttypen
30070 REM nebeneinander angezeigt.
32000 REM ----------------------------------------
32010 REM H. Voelz; 11.8.89
32020 REM XT-compatibler Rechner
```

Durch die Zeilen 1070 bis 1110 wird ein Zufallswort erzeugt. Durch eine spezielle Erzeugung der Zufallsdaten werden hier sowohl die Probleme des Zeichensatzes des Commodore als auch die Auswahl von nur Buchstaben gleichermaßen gelöst. Der Bereich der Zufallszahlen wird mit der Entscheidung IF RV < 0.5 geteilt und es werden getrennt große und kleine Buchstaben erzeugt. Dabei wird gerade wegen des Commodore von ASC("a") bzw. ASC("A") ausgegangen. Jedes so erzeugte Wort wird anschließend als zweites Wort mittels **GOSUB 330** in ein Wort geändert, das

nur Großbuchstaben enthält. Beide Varianten werden dann für die M = 21 Wörter (Zeile 1150) zur Kontrolle angezeigt.

Das Sortieren erfolgt nur bei den Wörtern mit Großbuchstaben. Zur Vermeidung der garbage collection erfolgt es indirekt über die Pointer in A(J). Verwendet wird ein verbesserter Bubble-Sort, welcher die bereits sortierten Wörter mit FOR J = 0 TO M-1-I (Zeile 1180) ausblendet und über das Flag F entscheidet, ob noch weiter sortiert werden muß. Die Anzahl der Sortierläufe wird angezeigt und anschließend das sortierte Feld.

Das Programm "FORMAT" arbeitet ebenfalls im Text-Mode und betrifft die richtige Formatierung von Zahlen (In einigen Dialekten existiert PRINT USING.). Sie sollen auf der Variablen SR stehen und werden dann mit **GOSUB 310** unter Berücksichtigung von zwei Parametern formatiert dargestellt:

CT (*Count Text*) enthält die Anzahl der darzustellenden Ziffern und Hilfszeichen (Gesamtlänge von SR$),

CN (*Count nach Komma*) benennt die davon hinter dem Dezimalpunkt stehenden Ziffern.

Je nach dem Rechnertyp sind hier beachtliche Leistungen, aber auch spezifische Rundungsfehler möglich. Deshalb wurde als Ausgangswert in SR$ eine sehr lange Ziffernfolge abgelegt. Sie wird schrittweise 14mal durch 9 dividiert und je als formatierte und für den Rechner übliche Zahl dargestellt. So wird diese Routine einschließlich ihrer Grenzen demonstriert. Das angezeigte Ergebnis hängt dabei auch von der Arithmetik des verwendeten Rechners ab.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### FORMAT ###
1Ø1Ø SR$="Formatieren": HO=INT((HO-LEN(SR$)-6)/2)
1Ø2Ø VE=Ø: GOSUB 11Ø: GOSUB 15Ø: PRINT
1Ø3Ø SR$="123456789123": INPUT "Zeichenzahl ="; CT
1Ø4Ø SR=VAL(SR$): INPUT "Nachkommastellen ="; CN
1Ø5Ø PRINT: PRINT "SR$";TAB(CT+5);"SR"
1Ø6Ø FOR I=1 TO 14
1Ø7Ø   GOSUB 31Ø: PRINT SR$; " : "; SR: SR=SR/9
1Ø8Ø NEXT I
1Ø9Ø INPUT "Ø=Ende; 1=weiter";B
11ØØ IF B=1 THEN 1Ø3Ø
111Ø GOTO 95Ø
112Ø REM ----Programm-Ende------
3ØØØØ REM Zeigt die Moeglichkeiten der Zahlen-
3ØØ1Ø REM darstellung. Der String ist bewusst
3ØØ2Ø REM so gross gewaehlt, dass auch
3ØØ3Ø REM die Genauigkeit der Arithmetik und
3ØØ4Ø REM die Rundung getestet werden kann.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 1Ø.5.89
32Ø2Ø REM XT-compatibler Rechner
```

2.4. Der Grafik-Mode

Für mehrere Anwender der Kleinstrechner erhebt sich hier sofort die Frage nach dem Warum eines speziellen Grafik-Mode. Sie sind es gewohnt, ohne Änderung sowohl Text als auch einzelne Punkte (Pixel) auf den Bildschirm zu übergeben. Aber bereits bei einigen leistungsfähigen BASIC-Dialekten existiert z. B. der Befehl SCREEN n. Je nach der Wahl von n kann ein Text oder Pixelformat ausgewählt werden, und die Grafik-Befehle sind nur im Grafik-Mode zulässig. Bei BASICODE sind

die Verhältnisse noch komplexer. Es soll ja schließlich ein Grafik-Programm auf Rechner mit sehr unterschiedlichen Bildschirmen immer die gleiche Grafik erzeugen. Der KC85/4 besitzt z. B. 320*256 Pixel. Der KC87 ermöglicht überhaupt keinen direkten Pixelzugriff. Er kennt nur eine Pseudografik. Andere Rechner haben mehr oder weniger Pixel als der KC85/4. Diese ganzen Verhältnisse muß der Bascoder bewältigen und unter der Berücksichtigung des jeweiligen Rechners in genau die eine Grafik umrechnen. Um dies zu ermöglichen und damit nicht den Textmode auch noch zu belasten, existiert der Grafik-Mode. Er wird durch die Subroutine **GOSUB 600** eingeschaltet. Ist dieser Zustand erreicht, so sind die meisten Befehle des Text-Mode nicht mehr zulässig. Ganz besonders sind dies die Subroutinen **GOSUB 110, 120, 150** und **220**, aber auch die BASIC-Befehle PRINT und INPUT. Sie sind erst dann wieder erlaubt, wenn mit **GOSUB 100** der Grafik-Mode abgeschaltet und wieder der Text-Mode erreicht ist. Dabei wird aber automatisch der Bildschirm gelöscht. Genau deshalb gibt es auch eine spezielle Subroutine, welche die Möglichkeit eröffnet, Text im Grafik-Mode zum Bildschirm zur Anzeige zu bringen. Hierauf wird aber erst im nächsten Programm eingegangen.

Wie schon erwähnt, existieren im Grafik-Mode für die einzelnen Rechnertypen sehr unterschiedliche Pixel-Anzahlen (vgl. Abschnitt 11.8.). Sie wurden ja schon mit dem Programm DATEN für ihren Rechner angezeigt. Damit nun ein Grafik-Programm auf allen Rechnern das gleichartige Bild erzeugt, müssen diese Unterschiede vom BASICODE automatisch berücksichtigt und passend verrechnet werden. Deshalb wurden spezielle Festlegungen für die Koordinaten des Bildschirms getroffen. Seine Breite und Höhe sind auf 1 normiert, wobei die Werte 1 für die x- bzw. y-Richtung bereits außerhalb des Bildschirmes liegen. Der x-y-Punkt 0, 0 liegt links oben sichtbar am Rande des Bildschirmes. Der Mitte des Bildschirms entsprechen die Werte 0.5, 0.5. Die jeweiligen Koordinatenwerte werden im Grafik-Mode mit den Variablen HG und VG übergeben. Das Längen-Höhen-Verhältnis ist auf genau 4:3 festgelegt. Durch diese Festlegung wird z. B. garantiert, daß ein Kreis auf jedem Rechner auch wirklich ein Kreis ist. Dies hat aber auch zur Folge, daß nicht immer alle Pixelwerte eines Rechners ausgenutzt werden können. Es empfiehlt sich daher, dies einmal mit dem Programm Daten zu überprüfen. Für ein Quadrat muß sich also z. B. HO um 0.3 und VE um 0.4 ändern. Die folgende Abbildung zeigt das Bildschirmformat:

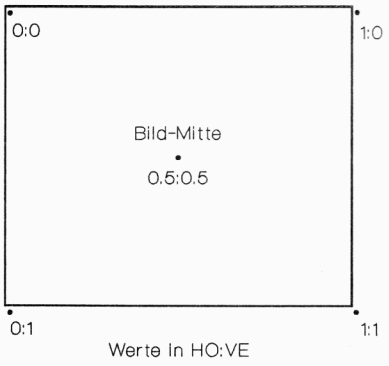


Abbildung 2.1. Punkte auf dem Grafikschirm

Im folgenden Programm wird nun mit diesen Mitteln ein kleines Bild gezeichnet. Die entsprechenden Punkt-Koordinaten sind in der DATA-Zeile 25010 abgelegt. Sie werden in die Felder X(I) und Y(I) übernommen. Mit den Daten in Zeile 25020 wird die Reihenfolge angegeben, mit der die Punkte "angelaufen" werden. Damit dies deutlich erkennbar ist, wurde eine Zeitverzögerung über **GOSUB 450** eingefügt. Der vorher an die Variable **SD** (*sound duration*) übergebene Wert bewirkt eine Pause von SD*0.1 Sekunden. Diese Pause kann durch Betätigen einer Taste vorzeitig beendet werden.

Mit dem Befehl **GOSUB 620** wird ein Punkt entsprechend den Werten in HO und VE gesetzt. Gleichzeitig liegt dann an dieser Stelle der unsichtbare Grafik-Cursor. Seine Koordinaten bleiben auch bei Veränderung von HO und VE im Rechner gespeichert und müssen für weitere Grafik-Befehle beachtet werden. Genau, wie der sichtbare Cursor im Text-Mode, bestimmt er nämlich die künftige Schreibposition für Texte bzw. für den Anfang einer Linie. Da er aber nicht sichtbar und auch nur bedingt - bis zur Veränderung von HG und VG - abrufbar ist, muß ihm beim Programmieren eine besondere Aufmerksamkeit gewidmet werden. Mit dem Befehl **GOSUB 630** wird vom Punkt des Grafik-Cursors eine Linie zum durch HO und VE bestimmten Punkt gezogen und der Grafik-Cursor auf diesen Endpunkt gesetzt.

Für die Grafik-Befehle kann über **CN** (*color numerisch*) auch ein "Farbwert" übergeben werden. CN = 0 bedeutet die Vordergrundfarbe (immer weiß) und CN = 1 schwarz (also die Hintergrundfarbe). Mit CN = 1 kann folglich ein Pixel oder eine Linie wieder gelöscht werden. Mehr als diese beiden "Farbwerte" sind im BASICODE-3 nicht vorhanden. Farbdarstellungen sind also noch nicht vorhanden. Aber die Variable CN zeigt dem Eingeweihten, daß zu einem späteren Zeitpunkt - wahrscheinlich mit BASICODE-4 - so etwas erfolgen wird. Einstweilen muß bei Farbgrafiken noch auf die Spezifika des jeweiligen Rechners zurückgegriffen werden. Dies sollte aber nur ausnahmsweise und dann in den Zeilen 20000 bis 24999 mit ausführlichen Kommentaren erfolgen (Zeilen ab 30000).

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### GRAFIK ###
1Ø1Ø DIM X(4), Y(4): RESTORE
1Ø2Ø FOR I=Ø TO 4 :READ X(I), Y(I): NEXT I
1Ø3Ø GOSUB 6ØØ: CN=Ø: GOSUB 25Ø
1Ø4Ø HO=X(Ø): VE=Y(Ø): GOSUB 62Ø
1Ø5Ø FOR I=Ø TO 7
1Ø6Ø   READ X: SD=5: GOSUB 45Ø
1Ø7Ø   HO=X(X): VE=Y(X): GOSUB 63Ø
1Ø8Ø NEXT I
1Ø9Ø GOSUB 25Ø: GOSUB 21Ø: GOTO 95Ø
25ØØØ REM---- Data-Zeilen ------
25Ø1Ø DATA .25, .75, .25, .3, .75, .3, .75, .75, .5, .1
25Ø2Ø DATA 1, 2, Ø, 3, 1, 4, 2, 3
3ØØØØ REM ---- Programm-Ende-----
3ØØ1Ø REM Es wird ein Bild gezeichnet.
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 11.5.89
32Ø2Ø REM XT-compatibler Rechner
```

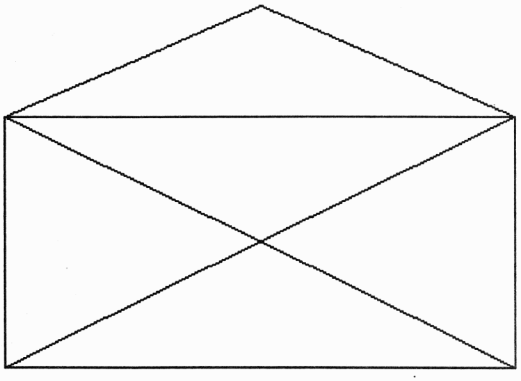


Abbildung 2.2. Grafik

Ein zweites Grafik-Programm zeigt bereits etwas genauer die Leistungsfähigkeit der Grafik-Routinen im BASICODE. Es wird das ineinander geschachtelte N-Eck gezeichnet. Mit den Eingabeparametern können Sie versuchen, die grafischen Grenzen Ihres Rechners zu erproben. Vielleicht gehen Sie dazu vom Dreieck mit einem Dreh-Winkel von 20 Grad aus und verkleinern den Winkel und/oder erhöhen die Eckenzahl. Bei der Wahl eines zu großen Dreh-Winkels vergrößert sich das N-Eck und es muß ein spezieller Abbruch erfolgen. Deshalb ist das AND in Zeile 1210 notwendig. Zur Unterstützung Ihres Gedächtnisses wird am Ende jedes Bildes der alte Winkel angezeigt. Dieser und der folgende Text bezüglich der Fortsetzung des Programms sind Texteingaben, die im Grafik-Mode mittels SR$ an die Routine **GOSUB 650** übergeben werden.

Schließlich sei noch auf die Klammern im Zusammenhang mit den logischen Befehlen hingewiesen. Gemäß Zeile 1040 und 1210 sind sie, um die logischen Ausdrücke, im Zusammenhang mit OR, AND und NOT in BASICODE (wiederum zur Kompatibilität bezüglich aller Rechner) notwendig. Generell sind deshalb die logischen Operatoren nur bezüglich Vergleichen zulässig. Unter anderem wird "wahr" nämlich in den verschiedenen Rechnern mit + 1 oder -1 dargestellt. Verboten ist folglich in BASICODE so etwas wie 7 AND 45.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### N-ECK ###
1Ø1Ø HO=INT((HO-LEN(SR$)-6)/2): VE=Ø
1Ø2Ø GOSUB 11Ø: SR$="N-ECK": GOSUB 15Ø: PRINT
1Ø3Ø INPUT "Anzahl der Ecken"; N: N=INT(N)
1Ø4Ø IF (N>12) OR (N<3) THEN PRINT "Fehler": GOTO 1Ø2Ø
1Ø5Ø Q=6.283185/N: DIM X(N), Y(N): XX=4/3: YY=.5
1Ø6Ø R=1/3: P2=1/15: INPUT"Drehwinkel =";W
1Ø7Ø D=W*.Ø174533: CN=Ø: REM pi/18Ø
1Ø8Ø S=SIN(D): C=COS(D): P=COS(Q): T=SIN(Q)
1Ø9Ø P4=T/(C*T+S-P*S): X(1)=P: Y(1)=T: GOSUB 6ØØ
11ØØ FOR I=1 TO N-1
111Ø    P=I*Q: X(I)=COS(P): Y(I)=SIN(P)
112Ø NEXT I
113Ø X(Ø)=1: Y(Ø)=Ø: X(N)=1: Y(N)=Ø
114Ø HO=YY+R*X(Ø): RV=XX*R: VE=YY-RV*Y(Ø): GOSUB 62Ø
115Ø FOR I=1 TO N
116Ø    HO=YY+R*X(I): VE=YY-RV*Y(I): GOSUB 63Ø
117Ø NEXT I
118Ø FOR I=Ø TO N
119Ø    P=X(I)*C-Y(I)*S: Y(I)=Y(I)*C+X(I)*S: X(I)=P
12ØØ NEXT I
121Ø R=R*P4: IF (R>P2) AND (R<.5) THEN 114Ø
122Ø HO=Ø: VE=.1: SR=W: GOSUB 3ØØ: SR$="Winkel = "+SR$
123Ø GOSUB 65Ø: VE=.2: SR$="1:weiter  Ø=Ende"
124Ø GOSUB 65Ø: GOSUB 21Ø: IF IN=49 THEN 1ØØØ
125Ø GOTO 95Ø
3ØØØØ REM Es wird ein geschachteltes N-Eck gezeichnet
3ØØ1Ø REM Durch den Winkel wird die Dichte bestimmt
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 11.5.89
32Ø2Ø REM XT-compatibler Rechner
```

```
Winkel = 3

1:weiter  0=Ende
```

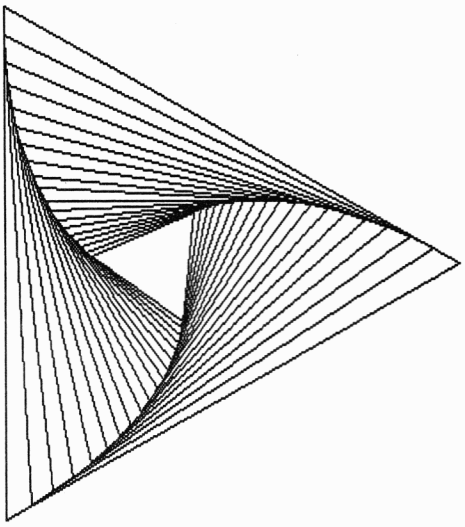
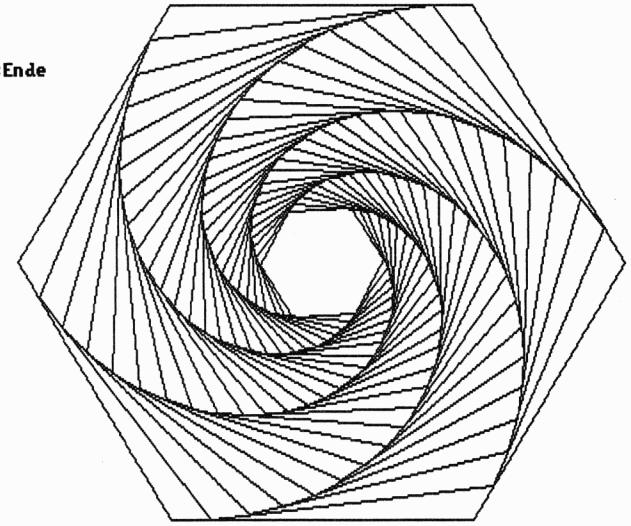


Abbildung 2.3. N-ECK = 3

```
Winkel = 7

1:weiter  0=Ende
```



Zwei besonders leistungsfähige Programme enthält das Kapitel 6.

2.5. Die Tonausgabe

Eine Tonausgabe - der Ton zur Aufmerksamkeit - wurde bereits mehrfach verwendet. Natürlich kann BASICODE etwas mehr, obwohl auch hier wieder auf die Grenzen der "schwächsten" Rechner Rücksicht genommen werden mußte. Die Tonausgabe ist daher nur auf einen Kanal beschränkt. Es kann also nicht mehrstimmig musiziert werden. Für den einen Ton kann aber über drei Parameter alles wesentliche erreicht werden:

SP (*sound pitch*)	die Tonhöhe in Halbtonabstufung mit 69 als Kammerton A (etwa 440 Hz).
SD (*sound duration*)	die Tondauer in Stufen zu 0.1 Sekunden.
SV (*sound volume*)	die Lautstärke mit 0 = Null und 15 = maximal.

Die Grenzen für die Tonhöhe hängen vom Rechnertyp ab. Aber in der Regel sind mehrere Oktaven möglich. Dennoch ist es wegen des Klanges sinnvoll, sich auf den mittleren Bereich zu beschränken. Aufgerufen wird die Tonausgabe mit **GOSUB 400**. Mit dem folgenden Programm wird ein einfaches Kinderlied gespielt und der zugehörige Text parallel auf dem Bildschirm angezeigt.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ### LIED ###
1Ø1Ø RESTORE: SR$="Alle meine Entchen": VE=Ø
1Ø2Ø HO=INT((HO-LEN(SR$)-6)/2): GOSUB 11Ø: GOSUB 15Ø: PRINT
1Ø3Ø FOR I=1 TO 27
1Ø4Ø    SV=15: READ A$, SP, SD
1Ø5Ø    IF SD<Ø THEN SD=-SD: PRINT
1Ø6Ø    GOSUB 4ØØ: PRINT A$;
1Ø7Ø    SD=1: SV=Ø: GOSUB 4ØØ
1Ø8Ø NEXT I
1Ø9Ø GOSUB 21Ø: GOTO 95Ø
25ØØØ REM---- Data-Zeilen ------
25Ø1Ø DATA "Al",67,3,"le",69,3," mei",71,3,"ne",72,3
25Ø2Ø DATA " Ent",74,6,"chen",74,6,"schwim",76,-3
25Ø3Ø DATA "men",76,3," auf",76,3," dem",76,3," See",74,9
25Ø4Ø DATA "schwim",76,-3,"men",76,3
25Ø5Ø DATA " auf",76,3," dem",76,3," See",74,9
25Ø6Ø DATA "Koepf",72,-3,"chen",72,3," in",72,3," das",72,3
25Ø7Ø DATA " Was",71,6,"ser",71,6,"Schwaenz",74,-3
25Ø8Ø DATA "chen",74,3," in",74,3," die",74,3," Hoeh",67,1Ø
3ØØØØ REM ---- Programm-Ende-----
3ØØ1Ø REM Ein Kinderlied wird gespielt
32ØØØ REM ----------------------
32Ø1Ø REM H. Voelz; 11.5.89
32Ø2Ø REM XT-compatibler Rechner
```

Auch dieses Programm ist vor allem für die Demonstration gedacht. Es ist so geschrieben, daß zunächst der Ton und dann die Silbe als Text erscheint. Einige empfinden dies als unangebracht. Vielleicht versuchen Sie zur Übung einmal, die Reihenfolge umzustellen. Ein leistungsfähiges Musik-Programm, was sich stark an W. A. Mozart anlehnt, enthält Kapitel 6.

2.6. Umgang mit Daten-Files

Viele Programme benötigen oder erzeugen zusätzliche Daten, die gesondert zu verwalten sind. Solche Datenmassive werden meist als File oder Datei bezeichnet. In vielen BASIC-Dialekten ist dies nur bezüglich Felder mit den Befehlen CSAVE* bzw. CLOAD* realisierbar. Seltener besteht die Möglichkeit, Dateien zu eröffnen (OPEN), um dann die Daten in einer Datei abzulegen oder aus ihr herauszuholen. Danach ist die Datei wieder zu schließen (CLOSE). Dieses Prinzip erscheint dem Unerfahrenen oft kompliziert. Es ist jedoch sehr effektiv und ermöglicht, beliebige Daten (nicht nur Felder) definiert zu verwalten. Es hat sich insbesondere im Umgang mit Disketten- und Festplattenspeichern herausgebildet. Hier sucht man auch nicht nach dem Ort, wo die Daten stehen oder abgelegt werden sollen, sondern nur nach dem Namen der Datei. Die Arbeit gestaltet sich also völlig anders als mit dem Kassettenrecorder, bei dem das Zählwerk für den Ort des File genutzt wird. Die weitere Buchführung muß man selber auf einem "Zettel" übernehmen.

Für BASICODE wurde von verschiedenen Möglichkeiten der Speicherung das Prinzip der sequentiellen Datei ausgewählt. Das Öffnen der Datei bewirkt die Bereitstellung eines entsprechenden Speicherplatzes, dessen Anfang mit dem Namen der Datei beschrieben bzw. darauf getestet wird. Deshalb muß der Name hierfür auf der Variablen **NF$** (*Name des File*) übergeben werden. Er darf maximal 7 Zeichen enthalten. Ferner muß mitgeteilt werden, ob in die Datei geschrieben oder aus ihr gelesen werden soll. Dies erfolgt durch den Wert der Variablen NF. Es bedeuten

NF gerade	(0, 2, 4, 6)	Lesen vom Speicher,
NF ungerade	(1, 3, 5, 7)	Schreiben in den Speicher.

Die vier verschiedenen Möglichkeiten wurden deshalb gewählt, weil an einem Rechner mehrere Speicher angeschlossen sein können, z. B. mehrere Kassettenrecorder und/oder Diskettenlaufwerke usw. BASICODE läßt also in dieser Variante bis zu vier unterschiedliche Speicher zu. Für die einfachen Rechner mit nur einem Speicher, z. B. dem Kassettenrecorder, genügen also die beiden Werte 0/1.

Zunächst sei die Speicherung von Daten auf einem externen Medium, z. B. auf Kassette, betrachtet. Mit einem NF$ zugewiesenen Namen und NF = 1 kann nun der Speichervorgang eingeleitet, also die Datei eröffnet werden. (Im folgenden Programm geschieht dies ab Zeile 1120.) Hierzu ist lediglich **GOSUB 500** aufzurufen. Dadurch wird der Rechner in den Mode zur Dateiarbeit (weiterhin im Textmode, Grafikmode ist aber zugelassen) versetzt. Anschließend werden Daten aus dem internen RAM des Rechners auf Kassette geschrieben. Hierzu werden die Daten als String an SR$ übergeben und dann mit der Routine **GOSUB 560** in den Speicher geschrieben. Dieser Schritt kann beliebig oft wiederholt werden. Endgültig, d. h. als Block mit Endezeichen, gespeichert sind diese Daten aber erst dann, wenn die Datei (das File) abgeschlossen wird. Dies erfolgt mit dem Befehl **GOSUB 580**. Dadurch wird auch wieder der normale Zustand des Rechners erreicht.

In analoger Weise werden die Daten aus dem Speicher geholt . Für die Routine **GOSUB 500** ist dazu lediglich NF = 0 (oder 2, 4, 6) zu wählen. Jetzt wird aber geprüft, ob der Name NF$ im File existiert. Wenn dies der Fall ist, wird von der Routine in IN der Wert 0 übergeben. Bei anderen Werten existiert das File nicht. Das Lesen erfolgt mit der Routine **GOSUB 540**, und es werden die String immer auf IN$ übergeben. Abschließend muß das File wieder mit **GOSUB 580** geschlossen werden.

Nach jeder File-Routine wird generell an IN ein Code übergeben, der darauf hinweist, ob der Befehl erfolgreich, d. h. fehlerfrei realisiert werden konnte. Es ist daher immer sinnvoll, IN abzufragen (dies wurde zur Vereinfachung des Programms im folgenden Beispiel weggelassen). Ist IN < > -1, so war alles in Ordnung.

Mit dem folgenden Programm können Zeichenketten, die z. B. einen Namen und die dazugehörende Telefon-Nr. enthalten, abgespeichert werden. Im Gegensatz zu CSAVE* und CLOAD*

brauchen dabei nur die wirklich belegten Feldelemente gespeichert zu werden. Das ist doch sehr vorteilhaft! Die Abfrage erfolgt später nach dem Anfang der Zeichenkette, wobei die Zeichenzahl frei wählbar ist.

Sie rufen also zunächst das Programm auf und starten es. Dann geben Sie die Namen und die zugehörigen Telefon-Nummern ein. Wenn Sie diese Arbeit abgeschlossen haben, geben Sie ENDE ein. Dadurch geht der Rechner in die Dateiarbeit über und verlangt den Namen für die Datei. Vielleicht wählen Sie TELEFON. Nun werden die Daten abgelegt (vorher Kassettenrecorder starten!). Wenn dies erfolgt ist, kehrt der Rechner in den normalen Mode zurück.

Er verlangt:

```
1:File erzeugen
2:File lesen.
```

Dieser Zustand wird aber auch beim Start des Programms erreicht, und so ist es möglich, ein vorhandenes File einzulesen.

```
1ØØØ A=5ØØØ: GOTO 2Ø: REM ### FILE ###
1Ø1Ø M=99: SR$="File-Arbeit": VE=Ø: DIM A$(M)
1Ø2Ø HO=INT((HO-LEN(SR$)-6)/2): GOSUB 11Ø: GOSUB 15Ø: PRINT
1Ø3Ø INPUT"1:File erzeugen 2:File lesen"; B
1Ø4Ø IF B=1 THEN 1Ø7Ø
1Ø5Ø IF B=2 THEN 117Ø
1Ø6Ø PRINT"falsche Eingabe": GOTO 1Ø3Ø
1Ø7Ø PRINT "Eingabe von Namen: Telefon-Nr."
1Ø8Ø PRINT"Beenden mit: ENDE"
1Ø9Ø FOR I=Ø TO M
11ØØ PRINT "Nummer:";I+1;" ";
1111   INPUT A$(I): IF A$(I)="ENDE" THEN N=I: I=M
112Ø NEXT I
113Ø GOSUB 25Ø: PRINT "speichern des File"
114Ø INPUT"Name =";NF$: NF=1: GOSUB 5ØØ
115Ø FOR I=Ø TO N: SR$=A$(I): GOSUB 56Ø: NEXT I
116Ø GOTO 121Ø
117Ø PRINT"Name =";: NF=Ø: INPUT NF$: GOSUB 5ØØ
118Ø FOR I=Ø TO M
119Ø   GOSUB 54Ø: A$(I)=IN$: IF IN$="END" THEN I=M
12ØØ NEXT I
121Ø GOSUB 58Ø: PRINT "Das File kann genutzt werden"
122Ø PRINT "gesuchter Anfang des Namens; Ø=Ende"
123Ø INPUT A$: L=LEN(A$): N=-1: IF A$="Ø" THEN GOTO 95Ø
124Ø FOR I=Ø TO M
125Ø   IF LEFT$(A$(I),L)=A$ THEN N=I: I=M
126Ø NEXT I
127Ø IF N<Ø THEN PRINT "nicht vorhanden": GOTO 122Ø
128Ø PRINT A$(N): GOTO 122Ø
3ØØØØ REM ---- Programm-Ende-----
3ØØ1Ø REM Es wird ein Datenfile erzeugt, in welchem
3ØØ2Ø REM gesucht werden kann. Es kann gerettet und
3ØØ3Ø REM geladen werden. Als Beispiel koennen hier
3ØØ4Ø REM Namen und Telefon-Nr. eingegeben werden.
3ØØ5Ø REM Beispiel: "Meyer 27 5Ø8 35"
3ØØ6Ø REM Gesucht wird nach dem Anfang der Zeichenkette.
```

```
30070 REM z. B. "Mey" oder "M" usw.
32000 REM ----------------------
32010 REM H. Voelz; 11.5.89
32020 REM XT-compatibler Rechner
```

2.7. Bemerkungen zum Bascoder

Der Bascoder ist das entscheidende Programm, welcher die Möglichkeiten von BASICODE zu realisieren gestattet. Er berücksichtigt also die Besonderheiten des jeweiligen Rechnertyps. Deshalb müssen soviel unterschiedliche Bascoder existieren wie Rechnertypen für BASICODE verwendet werden. Das Bascoder-Programm wird mit der üblichen rechnerspezifischen Art - meist über einen Kassettenrecorder - eingelesen. Hier kann also noch nicht das universelle BASICODE-Format benutzt werden. Es wird ja erst durch den Bascoder geschaffen.

Neben den verschiedenen zuvor besprochenen GOSUB-Routinen stellt der Bascoder also vor allem die neuen Schreib- und Lese-Routinen für das einheitliche BASICODE-Kassetteninterface bereit. Nur über dieses Interface ist ja der einheitliche Datenaustausch möglich. Seine Eigenschaften seien hier kurz genannt. Eine genauere Beschreibung existiert im Kapitel 4.

Bei diesem Interface wird die 0 durch eine 1200-Hz-Welle und die 1 durch zwei 2400-Hz-Wellen dargestellt. Die Datenübertragung erfolgt für jedes Byte asynchron mit einem Start- und zwei Stoppbit. Zusätzlich wird das achte Datenbit, also das höchstwertigste, invertiert. Dadurch stehen für die meisten Byte "eigentlich" drei Stoppbit bereit. Dies erhöht wesentlich die Übertragungssicherheit und ermöglicht insbesondere ein schnelles Einsynchronisieren nach einem Fehler. Durch diese Betriebsart ist BASICODE sehr zuverlässig. Auf Mittelwelle können so Entfernungen von mehr als tausend km überbrückt werden.

Die BASICODE-Programme werden im ASCII-Format, also nicht mit Token, gespeichert. Für die KC-Rechner entspricht dies etwa der Speicherung mit LIST#1 und dem Laden mit LOAD#1. Genau wie in dieser Betriebsart müssen die Programme dann zunächst in die interne Darstellung umgewandelt werden. Für alle in BASICODE gültigen BASIC-Befehle (Abschnitt 11.1.) realisiert dies der Bascoder. Hierfür werden u. a. Begriffe wie Übersetzen oder Translate verwendet. Für das Schreiben von BASICODE-Programmen auf Kassette erfolgt dann wieder die Rückwandlung in das ASCII-Format. Für die Speicherung sind weiter zwei Fälle zu unterscheiden. Normalerweise sollte nur das eigentliche BASICODE-Programm ab Zeile 1000 gespeichert werden. Es ist aber bei einigen Bascodern auch möglich, den Bascoder kombiniert mit dem Programm zu speichern. Dies sind dann alle BASIC-Zeilen; beginnend bei Zeile 10 bis zum Ende des BASICODE-Programms.

Der Befehl LIST hat meist im BASICODE die gleichen Eigenschaften wie im BASIC. Dies bedeutet, daß in der Regel mit LIST auch die BASIC-Zeilen Ihres Bascoders angezeigt werden. Für die Anzeige des BASICODE-Programms empfiehlt sich daher ein LIST 1000- . Diese Anzeige ist auch deshalb wichtig, damit Sie nicht eventuell versehentlich in Ihrem Bascoder editieren.

2.8. Ergänzungen zu den GOSUB-Routinen

Hier werden nur jene GOSUB-Routinen berücksichtigt, für die entsprechend den vorangegangenen Beschreibungen noch Ergänzungen notwendig sind bzw. dort nicht behandelt wurden. Eine vollständige, bewußt sehr kurz gehaltene Zusammenstellung enthält Abschnitt 11.3.

GOTO 20
Ist wegen GOTO eigentlich keine Subroutine. Dieser Befehl darf und muß einzig in Zeile 1000 stehen. Die übergebenen Werte für HO, VE sind wegen des Zählbeginns bei Null um eins kleiner als die entsprechenden Zeilen- und Spaltenzahlen. Die Rückkehr von dieser Initialisierungsroutine führt zur Zeile 1010. Während die Pixelzahlen weitgehend vom Rechnertyp abhängen und immer automatisch zu verrechnen sind, sollte man sich für die Textzeilen auf etwa 24 Zeilen und 40 Zeichen je Zeile beschränken. Noch günstiger ist es auch hier, die jeweiligen Werte des Rechners abzufragen und dann im Programm zu berücksichtigen.

GOSUB 110
Hier sollten immer die Maximalwerte von HO und VE berücksichtigt werden, um eine optimale Bildschirmdarstellung zu erreichen. Beispiele enthalten die Programme ZUTEXT in Zeile 1030 und TASTE in Zeile 1010. Die Werte von HO und VE werden durch den Aufruf nicht verändert.

GOSUB 150
Hierbei ist zu beachten, daß die Länge des String einschließlich der 2 mal 3 Leerzeichen nicht über das Zeilenende des Bildschirms hinausgeht.

GOSUB 200
Es sei noch einmal auf die rechnerspezifische Ausgabe in die Variablen IN und IN$ sowie auf den Unterschied für beide und die zulässigen Steuerzeichen hingewiesen. PRINT CHR$(IN) ist nicht sinnvoll. Die Besonderheiten der Commodore-Rechner müssen berücksichtigt werden.

GOSUB 220
Diese Routine liefert nur einen Wert in IN. IN$ wird also nicht verändert. In IN werden für Großbuchstaben 64 < = IN < = 96 und für Kleinbuchstaben 32 < = IN < = 64 übergeben. Wenn auf der Position des Bildschirmes ein Kleinbuchstabe steht, wird der Wert in IN den des zugehörigen Großbuchstabens annehmen. Werden Koordinaten außerhalb des Schirmes angesprochen, so ist IN = 0.

GOSUB 270
Sofern der Stringraum entsprechend organisiert ist, wird zunächst eine garbage collection ausgeführt. Dann werden die freien Speicherplätze (Byte) berechnet. Im Gegensatz zu FRE werden der Speicherplatz von FRE (X) und FRE (X$) addiert auf die Variable FR übergeben. Die kann z. B. dazu benutzt werden, ein Array maximal zu dimensionieren.

GOSUB 300
Die vergleichbare, in BASICODE nicht zugelassene Anweisung STR$ verhält sich bezüglich der Leerzeichen unterschiedlich. In dieser Routine werden alle Leerzeichen (Spaces) entfernt.

GOSUB 310
Wie das Beispielprogramm FORMAT demonstriert, kann der String durchaus mehr Stellen als die eigentliche Zahl enthalten. Bei einigen Rechnern werden dadurch die Eigenschaften der implementierten Arithmetik sichtbar. Bei der meist gebräuchlichen Binärarithmetik zeigen die zusätzlichen Stellen dann die Konvertierungsfehler beim Übergang von der Dezimalzahl (im ASCII- Code) an. Diese Fehler werden sonst meist nur bei der Subtraktion in der Umgebung von Null sichtbar.

GOSUB 350
Diese Routine wurde nicht besprochen. Sie wird nur im Zusammenhang mit Druckern verwendet und sendet den String SR$ an den Drucker. Initialisierungsroutinen für Drucker enthält BASICODE

in der Regel nicht. Es besteht eine gewisse Analogie zum unerlaubten BASIC-Befehl LPRINT bzw. PRINT#2.

GOSUB 360
Sendet Newline, also 0D, 0A (CRLF) an den Drucker.

GOSUB 400
Die Subroutine wird erst beendet, wenn der Ton abgelaufen ist. Es gelten folgende Datengrenzen, die aber nicht von jedem Rechner voll erfüllt werden:

SP (*Sound Pitch*)	für die Tonhöhe, 0 < = SP < = 127.
	Der Kammerton A liegt bei SP = 69, das zentrale C bei SP = 60.
	Eine ganze Zahl bedeutet einen Halbtonschritt.
SD (*Sound Duration*)	für die Dauer, 1 < = SD < = 255
SV (*Sound Volume*)	für die Stärke, 0 < = SV < = 15.
	SV = 0 bedeutet kein hörbarer Ton, Pause.

GOSUB 450
Der Abbruch dieser Routine kann auf zwei Arten erfolgen. Durch Ablauf der Zeit oder vorher durch Betätigen einer beliebigen Taste. Im ersten Fall gilt bei Rückkehr SD = 0, IN$ = "" und IN = 0. Im zweiten Fall enthält SD die noch verbliebene Restzeit und IN, IN$ die Werte gemäß GOSUB 200, 210. Für eine nicht unterbrechbare Warteroutine kann GOSUB 400 mit SV = 0 benutzt werden.

GOSUB 500
Die Routinen GOSUB 500, 540, 560 und 580 verwenden teilweise die unterste Bildschirmzeile für Anweisungen und Meldungen. Sie sind nur im Text-Mode nutzbar. Als Statusmeldung fungiert die Variable IN wie folgt:

IN = 0	:	Fehlerfreier Verlauf
IN = 1	:	Es wurde jetzt oder früher der letzte String gelesen
IN = -1	:	Ein nicht behebbarer Fehler ist aufgetreten.

Mit dem Code in NF werden vorbereitet:

NF = 0	:	Lesen von einer BASICODE-Kassette
NF = 1	:	Schreiben auf eine BASICODE-Kassette
NF = 2	:	Lesen von einem rechnerspezifischen externen Speicher
NF = 3	:	Schreiben auf einen rechnerspezifischen externen Speicher
NF = 4	:	Lesen von Diskette
NF = 5	:	Schreiben auf Diskette
NF = 6	:	Lesen von einer zweiten Diskette
NF = 7	:	Schreiben auf einer zweiten Diskette.

Nur für 0 und 1 wird universelle Lesbarkeit gemäß BASICODE bewirkt. Auf der untersten Bildschirmzeile können bei der File-Arbeit nützliche Hinweise erscheinen. Bei der Diskettenarbeit stellen die meisten Bascoder die Möglichkeit von bis zu drei geöffneten Files bereit.

GOSUB 540
Im Normalfall ist IN = 0, und IN$ enthält den gelesenen String. Bei Lesefehlern wird IN = -1. Der letzte String wird mit IN = 1 an IN$ übergeben. Bei weiteren Zugriffen bleibt IN = 1, und IN$ wird ein Leerstring. Wenn auf eine nicht geöffnete Datei zugegriffen wird, erfolgt eine Fehlermeldung.

GOSUB 600
Die Subroutinen GOSUB 110, 120, 150 und 220 sind verboten. Im Grafik-Mode ist auch keine File-Arbeit möglich. Der Grafik-Mode muß mit GOSUB 100 beendet werden.

GOSUB 620
Die unterschiedlichen Maßstäbe in X- und Y-Richtung von 4:3 und die Belegung mit dem Farbwert CN sind zu beachten.

GOSUB 630
Die Linie geht von der aktuellen (unsichtbaren) Cursor-Position zu den Pixel-Koordinaten HO, VE.

GOSUB 650
Der unsichtbare Cursor ist die linke obere Ecke des Schriftbeginns. Es ist darauf zu achten, daß die Schrift nicht den rechten Bildrand überschreitet. Die Schrift erscheint meist im normalen Textformat. Nach der Schriftausgabe ist der Ort des grafischen Cursors in HO und VE enthalten.

2.9. Hinweise zu den Variablen

Für die Variablen gibt es im BASICODE Einschränkungen, die in den Tabellen der Abschnitte 11.6. und 11.7. zusammengefaßt sind. Variablen in BASICODE bestehen aus ein oder zwei Zeichen, wovon das erste ein Großbuchstabe sein muß. Anfügen von %, ! oder # ist nicht erlaubt. Alle Variablen besitzen das Format real und je nach Rechnertyp zumindest 6 gültige Mantissenziffern. Stringvariablen sind durch Anhängen von $ gekennzeichnet.

Logische Variable werden indirekt realisiert und können nur "wahr" oder "falsch" sein. Mit logischen Variablen darf nicht numerisch gerechnet werden, denn "wahr" wird in Rechnern unterschiedlich durch +/- 1 oder andere Werte realisiert. Deshalb ist z. B. A = 3*(B = 1) nicht erlaubt. Statt dessen ist beispielsweise zu verwenden A = 0: IF B = 1 THEN A = 3.

Bevor eine Variable verwendet wird, muß ihr ein Wert zugewiesen werden. Nicht jeder Rechner setzt die Variablen am Programmbeginn zu Null. Für ein schnelles Programm ist auch die Reihenfolge dieser Initialisierung zu beachten. Oft gebrauchte Variablen sollten zuerst - eventuell auch mit Dummy-Werten - belegt werden (vgl. [7]). Dummy bedeutet Attrappe. Es wird also nur eine Wert-Belegung, ohne sie im Programm zu benutzen, verwendet.

Arrays sollten ebenfalls vor jeder Anwendung dimensioniert werden. Nicht alle Rechner wählen sonst automatisch den Wert 10.

3. Hinweise zur optimalen Strukturierung in BASICODE

In gewisser Hinsicht läßt sich jede Programmiersprache, also auch der Sprachteil vom BASICODE, mit einer natürlichen Sprache wie Englisch oder Russisch vergleichen. Die Befehle und speziellen GOSUB-Routinen entsprechen etwa den Vokabeln, die vorgeschriebene Syntax der Grammatik. Nun ist es zwar möglich, sich mit Vokabel- und Grammatik-Kenntnissen im Ausland zu verständigen. Um jedoch eine Fremdsprache zu beherrschen, muß man Sprachgefühl oder korrekter ausgedrückt Sprachkompetenz erwerben. Hieran wird der Muttersprachler bzw. der gute Dolmetscher erkannt. Genau in diesem Sinne habe ich in [7] den Begriff **Programmierkompetenz** eingeführt. Hierzu gehört auch ein guter **Programmierstil**.

Bei unterschiedlichen Fremdsprachen erwirbt man verschieden schnell Sprachkompetenz. So erreicht man beim Lernen von Englisch anfangs recht schnell Fortschritte. Jedoch ist es besonders schwer, Sprachkompetenz zu erwerben. Andererseits dauert es im Russischen oder Französischen relativ lange, bis man sich brauchbar verständigen kann. Ist dieser Stand aber erreicht, kommt man wesentlich schneller in Richtung Sprachkompetenz voran.

BASICODE ist in vielen Punkten BASIC-ähnlich, und BASIC ist jene Programmiersprache, die man zunächst sehr schnell erlernt, bei der es dann aber besonders schwierig ist, eine brauchbare Programmierkompetenz und einen guten Programmierstil zu erwerben, und genau deshalb ist BASIC bei einigen Experten mit gewissem Recht verrufen. Genau dies ist der Grund, warum dieses Kapitel von mir geschrieben wurde. Es stellt in einigen Punkten eine Kurzfassung von [7] dar, ist dann aber um Spezifika für BASICODE ergänzt und zum anderen wesentlich gezielter auf strukturiertes Programmieren ausgerichtet.

Allgemein ist zu beachten, daß Programmierkompetenz und guter Programmierstil kaum zu erlernen, sondern vor allem zu erwerben sind. Hierbei können Vorbilder, Beispiele, Regeln und Methoden nützlich sein. Ähnlich wie in der Kunst lernt man mehr im Umgang mit einem Meister oder an Meisterwerken, daß heißt hier aus hervorragend geschriebenen Programmen. Hoffentlich besitzen die Beispielprogramme dieses Buches etwas davon. Regeln sollen im folgenden aufgezeigt werden. Dabei gilt dann die Aussage: "keine Regel ohne Ausnahme".

Der Sprachteil von BASICODE besitzt (gemeinsam mit BASIC) folgende **Vorteile** gegenüber den meisten Hochsprachen:

- leichte Erlernbarkeit,
- hohe Interaktivität, insbesondere beim Programmieren,
- leichte Fehlererkennung und -behebung (gilt nur bezüglich des verwendeten Rechners, jedoch nicht für die Exaktheit der BASICODE-Programme),
- extrem wenig Vereinbarungen (in BASICODE ist der Gebrauch einiger zusätzlicher Variablen eingeschränkt).

BASICODE besitzt aber auch, wie viele ältere BASIC-Dialekte, deutliche **Schwächen**:

- keine Unterscheidung lokaler und globaler Variablen,
- keine Rekursivität,
- geringe Möglichkeiten zur Strukturierung,
- Gefahr des GOTO-Salates (Spaghetti-Code),
- keine dynamischen Arrays,
- eventuell größere Rechen- und Verarbeitungszeiten,
- Fehlen eines Compilers,
- Probleme der garbage collection bei den meisten Bascodern.

3.1. Einige Grundregeln der strukturierten Programmierung

Es gibt zumindest z. Z. keine einfachen und für jeden brauchbaren Methoden um zu prüfen oder gar zu beweisen, daß ein Programm **fehlerfrei** ist. Vielfach wird geglaubt, dies geschehe durch Tests. Ein **Test** beweist aber nur dann etwas, wenn er einen Fehler aufzeigt. Er beweist also Fehler, aber nicht die Abwesenheit von Fehlern. Selbst sehr viele Tests können nicht beweisen, daß ein Programm fehlerfrei ist. Ein Test - wie man ihn auch auslegt - durchläuft immer nur Zweige des Programms unter den gegebenen Bedingungen mit den gewählten Parametern. Bei Anwendung des Programms können folglich irgendwann einmal abweichende Fälle auftreten. Andererseits gibt es **Regeln**, wie man besonders **wirksame Tests** gestaltet. Dadurch erhält man eine höhere, aber keine absolute Sicherheit bezüglich Fehler. Hier seien zwei Grundregeln genannt:

- Jede Verzweigung sollte mindestens einmal und dazu mit extremen Werten durchlaufen werden.
- Bei Eingaben soll eine möglichst vollständige Überprüfung aller, auch der unsinnigsten Möglichkeiten erprobt werden.

Beides ist zumindest bei etwas umfangreicheren Programmen nahezu unmöglich. Bereits mit Mitteln der Mathematik ist sogar zu beweisen, daß es keine Möglichkeit gibt, um zu entscheiden, ob ein Programm das tut, was es tun soll. Ebenso ist es bereits theoretisch unmöglich, einen umfassenden Testdatensatz zu erstellen [13]. Dies sind nur einige Gründe dafür, daß allgemein gilt:

- Der Mensch ist der wichtigste Leser von Programmen!

Dies erscheint auf den ersten Blick unsinnig, denn schließlich soll ein Programm auf einem Rechner fehlerfrei laufen, und dabei braucht es kein Mensch zu lesen, sondern nur der Rechner zu interpretieren. Doch fehlerfrei kann es nur der Mensch machen. Deshalb gilt eine **wichtige Regel** für das Auffinden von Fehlern und Analysieren auf weitgehende **Fehlerfreiheit**:

- Das Programm wird von einem Autor geschrieben und von einem anderen kritisch gelesen. Hierzu muß er es "verstehen".

Doch dies ist nur ein Grund für die obige Aussage. Oft ist ein Programm zu ändern, weil sein Anwendungsgebiet sich weiter entwickelt hat oder ein Fehler erst später bemerkt wird. Auch hier muß das Programm erst wieder durch Lesen verstanden werden, bevor man Eingriffe vornehmen kann. Groß ist oft dabei der Aufwand, wenn das Programm schlecht geschrieben oder ungenügend dokumentiert wurde.

Hieraus folgt die **erste Forderung** für ein Programm:

Ein Programm muß immer gut lesbar geschrieben werden. Es sollten alle "Tricks" vermieden und, wenn sie schon unvermeidbar erscheinen, dann sollten sie zumindest sehr genau erklärt werden.

Aus den Erfahrungen der Geschichte der Rechentechnik konnten Erkenntnisse abgeleitet werden, die es gestatten, einige wesentliche Fehler mit nahezu vollständiger Sicherheit zu vermeiden. Sie haben sich in den Regeln der "strukturierten Programmierung" niedergeschlagen, u. a. [11] [12]. Hierbei hat N. Wirth entscheidend mitgearbeitet. Außerdem hat er auch die Entwicklung von PASCAL bestimmt. So ist PASCAL in den Ruf gekommen, schlechthin die strukturierte Sprache zu sein. Mittlerweile haben alle Sprachen - auch neuere BASIC-Varianten - hiervon profitiert, und es gibt auch bereits noch klarer strukturierte Sprachen als PASCAL, z. B. MODULA 2. Viel entscheidender ist aber die hierdurch allgemein bewirkte Änderung für das Programmieren. Ich vertrete den Standpunkt - und nicht nur ich allein - daß eine von der Sprache erzwungene Strukturierung oft nicht die beste Lösung ist. Der gute Programmierer braucht Freiheiten, die er im Sinne der strukturierten Programmierung einsetzen kann, aber nicht immer muß. Ein Korsett kann auch eine schöne Figur vortäuschen, und ein Keuschheitsgürtel ist keine Garantie für Treue!

Ein wichtiges Kriterium der strukturierten Programmierung ist der modulare Aufbau von Programmen. Hierfür gibt es mehrere Gründe:

Im Umgang mit hoher Komplexität hat fast jeder Mensch Probleme. Er kann eben nur ein Programm bestimmter Größe vollständig überblicken. Das sind etwa 10 bis 200 Befehle (Zeilen in BASICODE können weitaus mehr als einen Befehl enthalten!).

In einer Vielzahl von Programmen kommen sehr ähnliche bis gleiche Befehlsfolgen vor. Werden diese Abschnitte isoliert, so können sie ohne neues Entwerfen und Testen unmittelbar wiederverwendet werden. Dieses Modulkonzept ist bei BASICODE (wie bei den meisten BASIC-Varianten) leider nur durch Subroutinen zu realisieren.

Also gilt die **zweite Forderung:**

Ein Programm ist in Module (in BASICODE Subroutinen) so zu zerlegen, daß jeder Modul gut überblickbar ist, selbständig entworfen und erprobt werden kann und möglichst universell verwendbar ist.

Weitere Eigenschaften eines Moduls sind:

- Er hat eine genau definierte Aufgabe zu leisten.
- Er hat genau je einen einzigen Ein- und Austrittspunkt.
- Er besitzt eine genaue Dokumentation, zumindest bezüglich seiner Leistung sowie seiner Eingangs- und Ausgangsparameter, besonders bei BASICODE auch bezüglich der veränderten Variablen.

Die unübersichtliche Veränderung bei Variablen wird oft als "Seiteneffekt" bezeichnet. Deshalb wurden die globalen und lokalen Variablen eingeführt, die uns leider in BASICODE (wie in den meisten BASIC-Varianten) nicht zur Verfügung stehen. Daher ist ein besonders sorgfältiger Umgang mit **Variablen** einschließlich deren Bezeichnungen notwendig. Hierbei sind sorgfältig die maximal möglichen zwei Buchstaben auszuwählen und möglichst auch zu kommentieren. Im vorigen Kapitel wurde deshalb immer bei der Einführung der spezifischen Bascoder-Variablen ihre Bedeutung in Langform hinzugefügt. So läßt sich in der Regel ihr Sinn und Name besser merken.

Wenn nicht die Speicherkapazität überfordert wird, sollte kein Variablenname im Programm mehrfach in unterschiedlicher Bedeutung verwendet werden.

Ein zweiter wichtiger Punkt der strukturierten Programmierung wird oft fälschlicherweise und eingeengt mit dem **GOTO-Befehl** in Verbindung gebracht. Dabei wurde in den zentralen Arbeiten zur strukturierten Programmierung der GOTO-Befehl nicht einmal erwähnt, und selbst Wirth meint, daß an einigen Fällen ein GOTO-Befehl durchaus nützlich sein kann. Gefährlich ist es aber in jedem Fall, mit GOTO-Befehlen in einem Programm hin- und herzuspringen. Ein GOTO-Befehl darf folglich nur dann angewendet werden, wenn durch ihn nicht die Übersichtlichkeit leidet. Bei einem GOTO weiß man zwar immer, wo man hinspringt, es ist aber sehr schwer, den Weg zurück zu verfolgen. Allein deshalb sollten innerhalb eines Moduls sehr wenige GOTO vorkommen. Eine **tragfähige Empfehlung** lautet etwa wie folgt:

GOTO-Zeilen dürfen einander nur umfassen, aber nie kreuzen.

3.2. Grundstrukturen für Programme

Es ist ein Streit der Theoretiker, wieviel Programm-Strukturen mindestens notwendig sind, um alles (was prinzipiell berechenbar ist) zu programmieren. Wie der Maschinen-Code - insbesondere der leistungsfähigen RISC-Maschinen - zeigt, genügen sehr wenige. Es könnte durchaus sein, daß bereits zwei ausreichen:

- Die **Blockstruktur** und ihre einfache Aneinanderreihung zur Sequenz.
- Die **Entscheidung** mit dem bedingten Sprung als einfache Verzweigung.

Als dritte Möglichkeit werden vielfach noch **Schleifenkonstruktionen** gefordert. Sie bieten aber vor allem neue Möglichkeiten der strukturierten Programmierung und sind allein aus diesem Grunde vorteilhaft. Im folgenden werden diese Strukturen etwas ausführlicher betrachtet.

Die Blockstruktur

In BASICODE sieht ein sehr einfacher Block etwa wie folgt aus:

```
1Ø3Ø PRINT X
```

oder

```
1Ø4Ø C = (A + B) / 3
```

Eine Sequenz kann aus Mehrfachbefehlen in einer Zeile und/oder aus mehreren Zeilen bestehen:

```
1Ø5Ø PRINT"Berechnung der Summe zweier Zahlen": PRINT
1Ø6Ø INPUT"1. Zahl";A: INPUT"2. Zahl";B
1Ø7Ø PRINT A+B
```

Hier sind insgesamt 5 einfache Blöcke (Befehle) linear aneinandergefügt. Bildlich gilt:

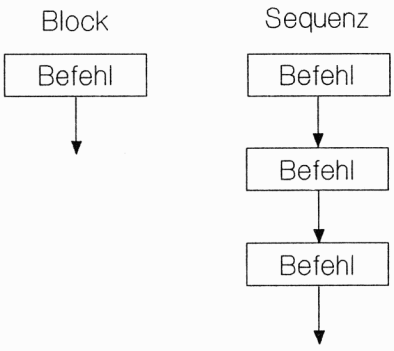


Abbildung 3.4

Die Verzweigungsstruktur

Die einfache Verzweigung existiert in der Form:

```
1Ø8Ø IF <Ausdruck> THEN <Anweisung|Zeilennummer>
```

Diese allgemeine (abstrakte) Beschreibung soll beispielhaft und konkret untersetzt werden:

Variante a:

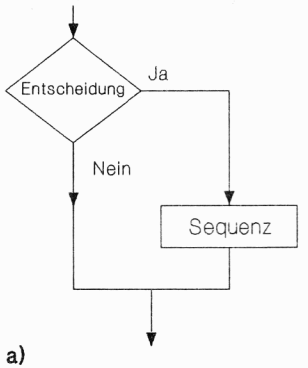
```
1Ø8Ø IF A=Ø THEN B=1: C=A*B: REM eventuelle weitere Befehle
1Ø9Ø ..... :REM nächsten gemeinsamen Befehle
```

bzw.

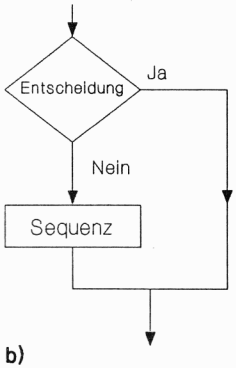
```
Variante b:

1Ø8Ø IF A=Ø THEN 11ØØ
1Ø9Ø ..... :REM einfache Befehle, Sequenz
11ØØ ..... :REM wieder gemeinsame Befehle
```

Der < Ausdruck > bedeutet also eine logische Entscheidung auf wahr oder falsch. In Betracht kommen dabei Vergleiche, wie A < B, A > = B; A$ > B$ usw. Das Ergebnis eines Vergleichs kann aber auch einer Variablen zugewiesen werden: V = A < B oder V = A$ < B$. Danach genügt es dann zu schreiben IF V THEN...

Die < Anweisung | Zeilennummer > bedeutet, daß die beiden Varianten möglich sind. Der "|" ist folglich als "oder" bezüglich der beiden Möglichkeiten zu verstehen. Dadurch ergeben sich in bildlicher Form die folgenden beiden Varianten



a)



b)

Abbildung 3.5

Die obigen BASICODE-Schreibweisen lassen die Bedeutung der strukturierten Programmierung, trotz der hinzugefügten Kommentare, nicht deutlich erkennen. Es ist nämlich wichtig, daß die beiden Äste der Verzweigung nur einen einzigen gemeinsamen Ausgang besitzen. Wenn dies nicht eingehalten wird, besteht die Gefahr des GOTO-Salates oder Spaghetti-Codes. Deshalb wurde bei den obigen Programmen auch jene Zeile bewußt hinzugefügt, ab der das Programm wieder im gemeinsamen Pfad ist, so wie es die Strukturbilder deutlicher zeigen. Aus ihnen ist weiter ersichtlich, daß die zusätzlich ausgeführte Befehlsfolge nur in einem Zweig liegt. Je nach der Wahl des IF ... THEN ... ist dies entweder der Zweig mit der erfüllten oder der unerfüllten Bedingung.

Oft werden mit den beiden Fällen der IF...THEN-Verzweigung sich nur etwas umständlich die Forderungen eines Algorithmus erfüllen lassen. Deshalb sollen weitere Möglichkeiten gezeigt werden.

Mit Hilfe des GOTO-Befehls gilt:

```
1Ø8Ø IF A=Ø THEN 111Ø: REM Sprung zur hierfür gültigen Befehlsfolge
1Ø9Ø ..........: REM einfache Befehle, Sequenz: ungültiger Zweig
11ØØ GOTO 112Ø  : REM Sprung zum gemeinsamen Ausgang
1111Ø ..........: REM einfache Befehle, Sequenz: gültiger Zweig
112Ø ..........: REM wieder gemeinsamer Ablauf
```

Obwohl hier sogar zwei Sprünge - der bedingte in Zeile 1080 und der unbedingte in Zeile 1100 - vorhanden sind, ist der strukturierten Programmierung Folge geleistet. Das zeigt das folgende Bild:

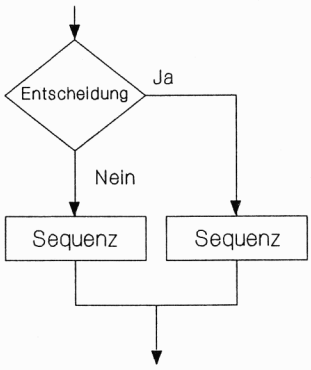


Abbildung 3.6

Es wird aber auch deutlich, daß beide Sprünge nicht zu große Zeilendifferenzen überbrücken sollten, also die Größe dieses Programmteils übersichtlich bleiben muß.

Natürlich gibt es noch eine Vielzahl anderer Lösungen in den Strukturen. Sie leiten sich meist durch Vereinfachung aus der letzten Struktur ab. Hier ein Beispiel:

```
1Ø8Ø IF A=Ø THEN B=1: C=D*B: GOTO 11ØØ
1Ø9Ø ..........: REM einfache Befehle, Sequenz: ungültiger Zweig
11ØØ ..........: REM wieder gemeinsamer Ablauf
```

Mit einer anderen Schreibweise lassen sich bei der Verzweigung sogar Sprünge vollständig vermeiden:

```
1Ø8Ø IF A=Ø  THEN  E=B: C=B*D: X=SIN(A): ......
1Ø9Ø IF A<>Ø THEN  E=C: C=B/D: X=COS(A): ......
11ØØ ..... : REM wieder gemeinsamer Zweig
```

Eine Verallgemeinerung dieses Prinzips ist die CASE-Struktur. In BASICODE also

```
11ØØ ON <Variable> GOTO <Zeilennummer, Zeilennummer,....>
```

bzw.

```
11ØØ ON <Variable> GOSUB <Zeilennummer, Zeilennummer,....>
1111Ø REM gemeinsamer Ausgang für alle Subroutinen
```

Die erste Struktur verführt geradezu dazu, für die einzelnen Ansprünge keinen gemeinsamen Ausgang zu realisieren. Bei der zweiten ist dies automatisch die Zeile 1110. Deshalb sollte sie, wo immer möglich, vorgezogen werden (siehe Kapitel 6.). Unabhängig von dem gemeinsamen Ausgang erzwingt diese Variante noch eine zweite Forderung der strukturierten Programmierung. Alle Subroutinen sind abgeschlossene Module und besitzen damit auch die o. g. Vorteile. Als Veranschaulichung kann Abbildung 3.5 gelten. Wobei für die GOSUB-Variante statt Sx (Sequenz x) in den Kästchen dann besser GOSUB x stände.

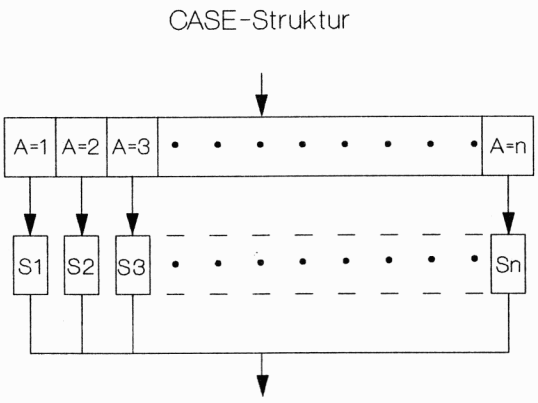


Abbildung 3.7

Die Schleifenstruktur

Die letzten Ergänzungen zur Verzweigungsstruktur ermöglichen in erster Linie eine bequemere Programmierung. Dasselbe gilt auch für die dritte Grundstruktur, nämlich die Schleife. Im BASICODE lautet sie:

```
11ØØ FOR X=<Startwert> TO <Endwert> [STEP <Schrittweite>]
1110     <Schleifenrumpf>
1120 NEXT X
```

Die eckigen Klammern bei STEP sollen aufzeigen, daß dieser Teil auch entfallen kann.

Der Schleifenrumpf ist in der Regel eine (unverzweigte) Sequenz. Er stellt einen Block im Sinne der strukturierten Programmierung dar. Seinem Anfang geht unmittelbar die FOR...NEXT-Befehlsfolge voraus. Auf seinen Ausgang folgt unmittelbar NEXT X. Nur innerhalb dieses Blockes sind Verzweigungen zugelassen. Ersichtlich darf aber nicht aus der Schleife heraus oder in die Schleife hinein gesprungen werden. Damit ist die FOR..NEXT-Konstruktion automatisch selbst ein Block im Sinne der strukturierten Programmierung. Dies bedeutet, daß der Schleifenrumpf wiederum eine Schleife - allerdings mit anderer Laufvariable, z. B. Y statt X - enthalten sein kann. Schleifen können in genau dieser Weise, aber nicht anders verschachtelt werden.

Bei Abarbeitung der Schleife ist es oft wichtig zu wissen, wie oft der Schleifenrumpf durchlaufen wird. In jedem Fall wird er einmal durchlaufen. Sofern nur ganzzahlige und positive Werte für die drei Parameter benutzt werden gilt:

Anzahl = (1 + Endwert - Startwert) / Schrittweite

Bei nicht ganzzahliger Schrittweite wird oft der Endwert nicht ganz erreicht. Generell können dann Rundungsfehler auftreten [7].

Die bisherige Variante des Schleifenkonstrukts bietet keine Möglichkeit, den Schleifendurchlauf vorzeitig, z. B. beim Eintreten einer bestimmten Bedingung abzubrechen. In vielen Sprachen wird hierzu die WHILE-WEND-Konstruktion verwendet. Sie kann in BASICODE zumindest auf zwei Arten realisiert werden: Über GOTO-Sprünge oder durch Änderungen in der Schleife. Die erste Variante ist vielleicht theoretisch "sauberer", hat aber den Nachteil, daß infolge der mehrfachen Zeilensuche längere Verarbeitungszeiten auftreten. Deshalb sei hier zunächst die zweite Variante besprochen. Bei ihr kann beispielsweise im Schleifenrumpf folgende Entscheidung eingefügt werden:

```
11ØØ FOR X=<Startwert> TO <Endwert> [STEP <Schrittweite>]
1110   <Schleifenrumpf1>
1120   IF <Bedingung> THEN X=<Endwert>: GOTO 114Ø
1130   <Schleifenrumpf2>
1140 NEXT X
```

Beim Erfülltsein der Bedingung wird die Laufvariable X innerhalb der Schleife auf ihren Endwert gesetzt und danach erfolgt ein Sprung zu NEXT X. Der Schleifenrumpf 2 wird dann also nicht mehr durchlaufen. Die Schleife wird daraufhin unmittelbar hinter NEXT X verlassen. Existiert nur der Schleifenrumpf 2, so ist es folglich auch möglich, ihn keinmal zu durchlaufen, nämlich dann, wenn beim Eintritt in die Schleife bereits die Bedingung erfüllt ist:

```
11ØØ FOR X = <Startwert> TO <Endwert> [STEP <Schrittweite>]
1110   IF <Bedingung> THEN X = <Endwert>: GOTO 114Ø
1130   <Schleifenrumpf>
1140 NEXT X
```

Oft ist es erwünscht, daß keine obere Begrenzung der Anzahl der Schleifendurchläufe feststeht. Dafür ist es in (fast?) allen BASICODE-Programmen möglich, die Schrittweite zu Null zu wählen.

```
11ØØ FOR X= <Startwert> TO <Endwert> STEP Ø
1110   <Schleifenrumpf1>
1120   IF <Bedingung> THEN X = <Endwert>: GOTO 114Ø
1130   <Schleifenrumpf2>
1140 NEXT X
```

Dann entscheidet ausschließlich die Bedingung über das Verlassen der Schleife. Dazu muß die Laufvariable X genau auf den < Endwert > gesetzt werden. Die Vielzahl der so möglichen Varianten zeigt die Nachteile der Realisierung von WHILE-WEND durch eine Schleife. Wenn also nicht die Ausführungszeit kritisch ist, sollte vielleicht doch das GOTO-Konstrukt bevorzugt werden:

```
11ØØ <Schleifenrumpf1>
1110 IF <Bedingung> THEN 114Ø
1120 <Schleifenrumpf2>
1130 GOTO 11ØØ
1140 REM: Ende
```

Ihm fehlt der Laufparameter X, und außerdem ist dieser Programmteil auch übersichtlicher.

Eine grafische Darstellung des allgemeinen Schleifenkonstrukts zeigt das folgende Bild:

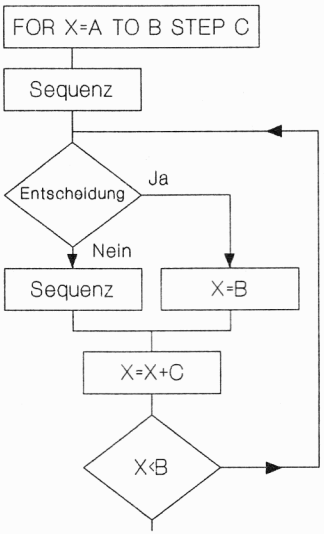


Abbildung 3.8

Es sei erwähnt, daß die hier benutzten Bilder oft den Namen PAP (Programmablaufplan) tragen. Eine andere Darstellungsmöglichkeit sind die Struktogramme. Auf sie wurde hier verzichtet.

3.3. Hinweise zur Fehlersuche bei BASICODE

In BASICODE existieren nur wenige spezielle Hilfsmittel zur Fehlersuche. In den meisten Fällen wird man auf die Methoden des speziellen Rechners zurückgreifen können. Generell sind in BASICODE deutlich zwei **Arten von Fehlern** zu unterscheiden:

- Fehler, welche auf dem verwendeten Rechner sichtbar werden,
- Fehler, welche ein Programm erst auf anderem Rechner zeigt.

Die Grenze zwischen diesen beiden Fehlerarten hängt stark vom jeweiligen Bascoder ab.

Die ersten Fehler sind unter den o. g. Einschränkungen weitgehend in Kommunikation mit dem Rechner und durch Tests auszuschalten.

Die zweite Fehlerart bereitet dagegen größere Schwierigkeiten. Sie läßt sich nur verringern, indem sich der Programm-Autor streng an die Regeln des BASICODE hält. Für einige Spezialrechner existieren hierzu zwar spezifische Testprogramme, sie sind jedoch der Allgemeinheit nicht zugänglich, existieren z. B. im Rundfunk der DDR.

Einige allgemeingültige Punkte für das Beheben von Fehlern

Zum Austesten von Programmen sollte man sich immer gut durchdachte Datensätze zusammenstellen. Sie legt man am besten in zusätzlichen und zeitweiligen DATA-Zeilen ab. Dann ersetzt man vorübergehend die INPUT-Anweisungen durch READ-Befehle. So umgeht man das wiederholte und leicht fehlerbehaftete Eingeben der Testdaten.

Oft bereitet das Eingrenzen auftretender Fehler erhebliche Schwierigkeiten. Hier können die folgenden Hinweise vielleicht nützlich sein. Zusätzlich zeitweilig eingefügte PRINT-Anweisungen können dazu benutzt werden, um auszutesten, ob diese Stellen überhaupt erreicht werden und welche Werte hier die angezeigten Variablen besitzen.

Da in BASICODE einerseits END und STOP nicht zulässig sind und andererseits GOTO 950 die Variablen z. T. beeinflußt, eignet sich folgender Weg zur Programmunterbrechung. Mit einem zusätzlichen zeitweiligen INPUT wird der Programmablauf angehalten. Durch unzulässige Eingaben oder Betätigen von BRK, Control-C oder einer entsprechenden Taste wird das Programm verlassen, und nun kann man sich die Variablen anschauen, sie Verändern und das Programm dann mit CONT weiterlaufen lassen. Auch ein GOTO zur unterbrochenen Zeile ist meist möglich.

Eine typische BASICODE-Methode ist ein zeitweiliges Einfügen von GOSUB 210. Generell kann ein zeitweiliges Umleiten über GOSUB nützlich sein. So lassen sich auch Teilprotokolle zum Drucker umleiten.

Schließlich sei erwähnt, daß Befehle durch ein zeitweilig davor gesetztes REM leicht auszuschalten und durch Löschen des REM wieder einzuschalten sind. So kann geprüft werden, ob und wie solche Befehle bzw. Zeilen wirken.

3.4. Bemerkungen zur garbage collection

Die Mehrzahl der Rechner verwalten die String auf eine völlig andere Art als die numerischen Variablen oder Felder. Ein String kann nämlich im Gegensatz zu einer numerischen Variablen zwischen 0 und 255 Zeichen lang sein. Daher ist es in der Regel nicht möglich, den gerade aktuellen String an jener Stelle im Speicher abzulegen, wo sein Vorgänger stand. Statt dessen werden die String in der Reihenfolge ihres Auftretens meist einfach hintereinander im Stringraum abgelegt. Pointer verweisen dann auf die jeweils aktuelle Adresse und Länge der String. Mit der Schleife

```
1Ø1Ø FOR X=1 TO 5: INPUT A$: NEXT X
```

werden z. B. der Reihe nach "Haus", "Schornstein", "Kleid", "Katze", "Lied" im Speicherraum für die String abgelegt. Dort ist dann zu finden: "HausSchornsteinKleidKatzeLied" und nicht, wie man bei einem eventuell auch denkbaren Überschreiben erwarten könnte: "Liedenstein"

Die erste Variante läßt sich nämlich technisch einfacher realisieren. Sie hat aber den Nachteil, daß im obigen Fall der gesamte Speicherplatz bis Katze mit nicht mehr gültigen String belegt ist. Sie können sich bestimmt vorstellen, daß bei häufiger Übergabe von String auf ein A$ bei dieser Methode irgendwann einmal der Stringraum völlig belegt ist und dazu noch mit fast nur überflüssigen, weil nicht mehr gültigen Zeichenketten. Man kann diese als Müll, Abfall oder englisch mit garbage bezeichnen. Der Rechner muß dann eine Routine aufrufen, die diesen Müll beseitigt, so daß danach nur noch die aktuellen String existieren. Dieser Vorgang heißt garbage collection (Müll-Beseitigung). Sie erfordert Zeit. In [7] wurde für den KC 85/2 bis KC 85/4 folgender Zusammenhang mit der Anzahl jener Stringvariablen, die in einem Programm existieren, gefunden. Er gilt ähnlich auch für andere Rechner, sofern sie eine Stringverwaltung nach dieser Methode verwirklichen:

Anzahl der benutzten String:	50	100	250	500	1000
Dauer in Sekunden:	0.4	1.7	11	41	165

Die Zeitdauer der garbage collection wächst also etwa quadratisch mit der Stringanzahl. Sie wird ab hundert deutlich spürbar und ab etwa 500 unerträglich. Dies ist ein Grund dafür, warum Textverarbeitung und ähnliches in BASICODE (BASIC und weiteren Sprachen) sehr umsichtig programmiert werden muß. Man muß sich bemühen, mit möglichst wenig Stringvariablen, d. h. auch mit klein dimensionierten Stringfeldern, auszukommen.

Es ist zu beachten, daß sich ein Rechner während der garbage collection wie defekt verhält. Unter anderem nimmt er keine Tasteneingabe entgegen.

Neben der **Dauer einer garbage collection** ist es auch wichtig, daß der **Abstand zwischen zwei** garbage collection so groß wie möglich gemacht wird. Dies läßt sich auf zweierlei Weise erreichen:

- Ist der Stringraum ausreichend groß definiert (mit A in Zeile 1000), so dauert es auch relativ lange, bis alle Reserven ausgeschöpft sind und die nächste garbage collection notwendig wird.
- Wenn man sich bemüht, möglichst selten String zuzuweisen, dauert es ebenfalls länger, bis die Reserven erschöpft sind.

Beide Methoden sollte man sinnvoll und mit Umsicht immer dann nutzen, wenn viele String im Programm sich nicht vermeiden lassen.

3.5. Optimierung von Programmen

Wenn von Optimieren gesprochen wird, liegen meist mehrere Wünsche vor, die nur teilweise gleichzeitig zu erfüllen sind. Ein Taschenradio soll z. B. klein und leicht sein, wenig Strom verbrauchen und eine gute Klangqualität besitzen. Da nicht alles zugleich zu erfüllen ist, sind Entscheidungen zugunsten einiger und zuungunsten anderer Parameter notwendig. Es muß ein Kompromiß eingegangen werden. Das entsprechende Verfahren nennt man Optimierung. Was bedeutet nun Optimierung beim Programmieren? Welches sind die wichtigsten Eigenschaften eines Programms, die hier beachtet werden müssen? Dies sind vor allem:

- Die **Geschwindigkeit**, d. h. die Zeit für die Abarbeitung des Programms.
- Der benötigte **Speicherplatz**.
- Die **Lesbarkeit** des Programms (siehe oben).

Die Wichtung dieser drei Punkte hängt natürlich vom Anwendungsgebiet ab. So spricht heute eigentlich kein Grund mehr dagegen, den vorhandenen Speicherplatz auszuschöpfen. Doch wieviel Speicherplatz besitzt der kleinste Computer mit BASICODE, auf dem unser Programm noch laufen soll? Das ist eine gewiß nicht einfach zu beantwortende Frage, insbesondere für komplizierte, d. h. lange Programme. Dabei sind folgende **Speicheranforderungen** zu unterscheiden:

- Platz für das **Programm** selbst,
- Statischer Platz für die **Variablen, Felder** usw.,
- Dynamischer Platz u. a. für den **Stack**,
- Platz für die **String**.

Bezüglich der Zeit ist es ein sehr großer Unterschied, ob man 1 Sekunde oder 1 Stunde wartet. Bei der Berechnung von Fraktalen, die durchaus Tage dauern kann, wäre bereits eine Verdopplung der Geschwindigkeit ein bedeutender Fortschritt. Bei online-Steuerungen, die wohl noch nicht bei BASICODE-3 vorkommen dürften, könnten dagegen schon Sekunden viel zu lang sein.
Als Richtwerte für den Zeitverbrauch einer arithmetischen Operation können wieder Ergebnisse vom KC 85/3 dienen [7]:

Operation:	+	-	*	/	^	SIN	LOG	EXP	SQR
Zeit (ms):	0.7	0.7	2.0	5.0	46	23	20	27	46

Bezüglich der Lesbarkeit von Programmen wurde auf die Bedeutung einer guten Selbstdokumentation bereits oben hingewiesen. Sie kann es sinnvoll machen, zwei Programmvarianten zu verwenden, eine mit und eine ohne Dokumentation. Andererseits ist hier BASICODE bereits vom Prinzip her ideal angelegt, indem ein großer Teil der Kommentare ohnehin ans Ende befohlen wird und dieses im Bedarfsfall leicht abzuschneiden ist.

Berücksichtigen wir die Spezifika von BASICODE und die Forderungen der strukturierten Programmierung, so kommen in Anlehnung an [7] (dort befinden sich auch detailliertere Erklärungen) insbesondere die folgenden Gesichtspunkte für die unterschiedlichen Optimierungen in Betracht.

Für eine schnelle Abarbeitung sollte beachtet werden:

- FOR...TO...NEXT statt GOTO und GOSUB verwenden.
- Mit GOSUB zu möglichst niedrigen Zeilenzahlen springen, d. h. Unterprogramme an den Anfang des Programms legen.
- GOTO minimieren.
- Variablen in der richtigen Reihenfolge initialisieren.
- Mehrfach verwendete Zahlen auf Variablen übertragen.
- Den Rumpf von FOR-NEXT-Schleifen, besonders den der inneren, einfach gestalten.
- REM weglassen oder an das Ende des Programms legen.
- Keine Rechnungen unnötig wiederholen.
- Häufiger benutzte Zahlenwerte auf Variablen übertragen, um die Konvertierungszeit ins interne Format zu unterdrücken.
- Arithmetik einfach gestalten, z. B.2*A -> A+A; A^2 -> A*A; multiplizieren statt dividieren.
- Logik aufgliedern bzw. indirekte logische Variablen verwenden.
- Häufig benutzte Feldelemente vorübergehend Variablen zuweisen.
- Wertetabellen für wiederholte Rechnungen anlegen.
- String möglichst selten aufrufen.
- Sortieren oder andere Stringaustausche über Pointer realisieren.
- Mit GOSUB 270 garbage collection beeinflussen.
- Stringraum optimal wählen.

Für die Minimierung des Speicherplatzes gilt:

a) beim Programmspeicher:

- Vielfachbefehle in einer Zeile verwenden.
- Kurze Variablennamen nutzen.
- Unterprogramme realisieren.
- Keine REM-Texte einfügen.
- Konstanten und String in Variablen bzw. DATA-Zeilen ablegen.
- Bei Argumenten von FOR, TO, STEP, PRINT usw. sind eventuell komplizierte Ausdrücke zu verwenden.
- LET weglassen.

b) für den statischen Programmablauf:

- Felder statt Variablen verwenden.
- Höhere Dimensionszahl der Felder statt mehrere Felder nutzen.
- Feldelement 0 verwenden.
- Felder nicht zu groß dimensionieren.
- Mit A = n in Zeile 1000 den Stringraum passend wählen.

c) für den dynamischen Programmablauf kann die folgende Stackbelastung als Richtwert gelten:

- FOR...NEXT	20 Byte
- GOSUB	5 Byte
- Offene Klammer	4 Byte
- Zwischenergebnis	12 Byte

Zur Begrenzung der Stacktiefe sind folglich nicht unnötig viele derartige Operationen ineinander zu schachteln.

d) für den dynamischen Stringaufruf:

- Den Speicherraum mit A = n in Zeile 1000 richtig festlegen.
- Günstiges Verhalten der garbage collection verwirklichen.

Für gut lesbare Programme gilt:
a) REM u. a. für folgende Erklärungen und Hinweise zu verwenden:

- Zweck des Programms,
- Bedeutung der Variablen,
- Programmteile wie Subroutinen, Tricks usw.,
- Literaturhinweise,
- Datum, Version, Copyright.
- Verwendeter Rechnertyp, evtl. Ausstattung

b) Optische Gliederung anwenden:

- FOR...TO...STEP auf einer eigenen Zeile anordnen.
- Dito NEXT.
- IF möglichst an Zeilenbeginn legen.
- Programmteile durch REM trennen.

Hinweis:
Die in diesem Buch bei der Programmdokumentation verwendeten zusätzlichen **Space** zur noch deutlicheren Strukturierung gehen beim BASICODE leider verloren.

4. Das Kassetteninterface von BASICODE

Eine herausragende Eigenschaft von BASICODE ist sein standardisiertes Datenformat. Es ist so ausgelegt, daß es vielfältige Anwendungen zuläßt. Ursprünglich war es nur für störfreie Rundfunkübertragungen und fehlerarme Kassettenspeicherungen entwickelt. Inzwischen hat sich aber gezeigt, daß es u. a. auch bedeutsame Möglichkeiten per Telefon und für Funkamateure bietet. Es hat den Anschein, daß die Festlegungen des Datenformats generell für den Daten- und Programmaustausch von Bedeutung sind. Der Grund hierin liegt vor allem in zwei Tatsachen:

- Die Kassette ist ein äußerst preiswertes Speichermedium und ein Kassenrecorder ist relativ leicht an alle Geräte anschließbar. Der hierfür notwendige Zusatzaufwand bleibt selbst in den ungünstigsten Fällen klein. Deshalb kann es nach diesem Prinzip sogar vorteilhaft sein, Daten nicht nur bei der 8-Bit-Technik, sondern auch bis zur 16- und 32-Bit-Technik auszutauschen. Eine Beschränkung auf BASIC ist hierbei ebenfalls nicht Bedingung, denn es kommt ja ohnehin meist der ASCII-Code zur Anwendung. So können beliebige Text-, also Quell-Files aller Sprachen ausgetauscht und übertragen werden.
- Der notwendige Frequenzbereich von etwa 400 bis 3600 Hz ist über fast alle existierenden Kanäle ohne Schwierigkeiten zu realisieren. Sein einziger Nachteil ist die relativ geringe Datenrate von 1200 Baud, also 1200 bit/s. Hier entsteht eine Begrenzung für große Files3.

Beim BASICODE-Übertragungsprotokoll werden zur Erhöhung der Störsicherheit zwei Signalformen und bis zu vier hierarchisch gestaffelte Informationsebenen der Datenstrukturierung unterschieden. Dies bedeutet, daß ein BASICODE-Protokoll nicht ganz einfach aufgebaut ist. Für die Nur-Anwender von BASICODE sind entsprechende Kenntnisse und damit dieses Kapitel nahezu uninteressant, denn sie haben praktisch nichts mit derartigen Problemen zu tun. Für sie löst alles der vorhandene Bascoder. Für viele andere - und das sind nicht nur neugierige Techniker - ist die folgende Beschreibung jedoch aus mehreren Gründen vorteilhaft, ja notwendig. Der Fachmann bekommt so einen Einblick in weitere Details, die ihm bei eventuell auftretenden Störungen recht nützlich sein können. Für Anwendungen außerhalb von BASICODE ist dieses Wissen geradezu unabdingbare Voraussetzung. Dasselbe gilt selbstverständlich auch für die Entwickler von neuen Bascodern oder für jene Fachleute, die nur die Kopplung zu unterschiedlichen Rechnern auf der Basis des BASICODE-Protokolls herstellen wollen.

Die beiden verwendeten, unterschiedlichen Filetypen sind:

- ASCII-Files, z. B. BASICODE-Programme
- Binär-Files, z. B. Daten.

Die ASCII-Files enthalten fast nur Daten der dezimalen Code-Werte 32 bis 127. Hierdurch ist es möglich, leicht Fehler zu erkennen. Außerdem können ausgewählte niedrigere oder höhere Codes als Sonderzeichen z. B. für den Start und das Ende des Files definiert werden. Deshalb werden für ein ASCII-File keine zusätzlichen Längenangaben benötigt. Es kann eine beliebige Länge besitzen. Bei BASICODE-Programmen wird daher auch stets das ganze Programm als eine fest zusammenhängende Struktur übertragen.

Bei Binär-Files sind alle Werte von 0 bis 255 als Daten zugelassen. Es kann also kein Sonderzeichen für den Anfang und das Ende definiert werden. Daher wird hier eine Blockstruktur genau gegebener Länge notwendig. Im BASICODE-Protokoll sind 1024 Byte für die Daten und vier zusätzliche Byte für Zusatzinformationen festgelegt. Jede zu übertragende bzw. zu speichernde Datei (File) muß folglich in derartige Längen zergliedert werden. Hierauf wird noch eingegangen.

Die vier hierarchisch gestaffelten Informationsebenen des BASICODE-Protokolls sind:

- Das Bit-Niveau.
- Das asynchrone Byte-Niveau.
- Der Datenblock bzw. ein ASCII-File oder BASICODE-Programm.
- Ein vollständiges Daten-File, welches aus einem oder mehreren Daten-Blöcken besteht.

Die sich so ergebenden vielen Möglichkeiten sollen nun der Reihe nach behandelt werden.

4.1. Bit-Niveau

Das Problem der Daten-Speicherung bzw. -Übertragung zeigt wesentliche Probleme vor allem auf dem untersten Niveau, dem Bit-Niveau. Genau deshalb sind hier ausführlichere Erörterungen notwendig. Während hier nämlich wesentlich die Eigenschaften des analogen Kanals mit all seinen Komplikationen eingehen, sind auf den höheren Niveaus nur noch digitale Betrachtungen notwendig. Auf dem Bit-Niveau sind drei unterschiedliche Kanalabschnitte wesentlich:

- Im Rechner liegen alle Daten (L- und H-Signale bzw. 0/1-Signale) in der Regel mit den Standard-TTL-Pegeln, also etwa 0.05 Volt und 4 Volt vor.
- Der Übertragungs- bzw. Speicherkanal ist in der Regel ein typischer Analog-Kanal mit Störungen.
- An den Rechner müssen letztlich wieder digitale L- und H-Signale übergeben werden, denn nur die kann er verwenden.

Hieraus leiten sich zwei Problemstellungen ab:

- Die digitalen Signale müssen so verändert (codiert) werden, daß sie möglichst fehlerarm den Analogkanal passieren können.
- Während es kein Problem ist, die Codierung zu erzeugen, ist es in der Regel recht schwierig, aus den gestörten Analogsignalen wieder die Digitalsignale zu erhalten.

Infolge dieser Doppelseitigkeit der Codierung - Anpassung auf den Analogkanal und Detektierung der Analogsignale bezüglich ihres digitalen Inhalts - ist es recht schwierig, eine optimale Codierung zu finden. Die Entscheidung betrifft sowohl Hardwarestrukturen (vor allem die Detektorschaltung) als auch Softwareaufwand.

Hier kann nur die Lösung im BASICODE-Protokoll mit ihren Vor- und Nachteilen behandelt werden. Sie betrifft bevorzugt den Rundfunk- und den Tonbandkanal. Für ihn gelten etwa die folgenden Eigenschaften:

- Es ist keine Übertragung von Gleichstromwerten möglich.
- Der Störabstand ist oft nicht größer als 40 dB.
- Der Frequenzbereich reicht maximal von 30 bis 15000 Hz (Mittelwelle 4500 Hz).
- Es können beachtliche Phasenfehler vorkommen.

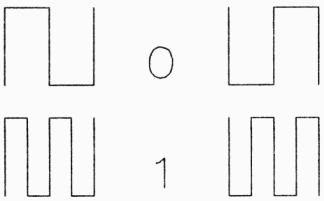


Abbildung 4.9.

Aus dieser Sicht ist die Codierung der 0 (L) mit einer 1200 Hz-Rechteckschwingung und der 1 (H) mit zwei 2400 Hz-Schwingungen recht günstig, denn dieses Prinzip hat mehrere Vorteile:

- Beide Signale sind gleichlang, dies vereinfacht die Detektion und Decodierung.
- Beide Signale sind gleichstromfrei.
- Die Phasenlage der Signale ist ohne Bedeutung. Beide im Bild vorgestellten Möglichkeiten sind zulässig. (Jedoch bei einer Übertragung oder Speicherung nur die linke oder rechte Variante, jedoch nicht beide gleichzeitig.) Es braucht u.a. nicht darauf geachtet zu werden, wieviel Verstärkerstufen im Prozeß existieren.
- Es genügt eine Bandbreite von etwa 400 bis 3600 Hz zur Übertragung. Sie wird selbst von recht mäßigen Kanälen, wie AM- Rundfunk, Telefon usw., erreicht.

Die Erzeugung der beiden Signale (1200 Hz und 2400 Hz) ist recht einfach und bedarf daher keiner Erörterung. Beide enthalten als Rechteckschwingung zwar ein unendliches Spektrum. Von den 1200 Hz muß aber nur die dritte Harmonische mit merklicher Amplitude existieren. Von der 2400 Hz-Schwingung wird keine Oberwelle mehr benötigt. So entsteht bei der Wiedergabe bzw. am Ausgang des Rundfunkempfängers ein mehr oder weniger stark verändertes Signal. Dieses Signal ist nicht mehr digital und enthält unterschiedliche Amplituden und eine deutlich veränderte Kurvenform. Hinzu kommen noch die Einflüsse des Kanals, wie Amplitudenschwankungen, Störsignale (z. B. Rauschen), Phasenschwankungen und Nichtlinearitäten. Damit wird deutlich, daß an die Signaldetektion trotz allem noch beachtliche Forderungen zu stellen sind. Sie sind aber deutlich geringer als bei anderen Kassettenverfahren. Es müssen also sowohl günstige Schaltungen als auch Rechenprogramme gefunden werden, die wieder die originalen L- und H-Werte herstellen. Sowohl theoretisch als auch praktisch ist das hier gewählte Prinzip bezüglich der Störsicherheit stark im Vorteil. Es erhebt sich dabei natürlich die Frage, warum es dann nicht grundsätzlich und ausschließlich genutzt wird, zumal es schon über zwanzig Jahre bekannt ist. Es besitzt eben auch Nachteile: Es ist nicht das schnellste Verfahren und erreicht damit auch nicht die bestmögliche Speicherdichte bzw. Übertragungsrate. Auf Kosten der Sicherheit würde z. B. auch eine 2400 Hz-Schwingung für die 1 genügen. Dann könnte man im Mittel die anderthalbfache Übertragungsrate bei etwa gleicher Bandbreite erreichen oder die anderthalbfache Menge Information auf einer Kassette speichern. Durch weitere Methoden läßt sich die Datenrate bzw. Speicherdichte noch wesentlich weiter steigern. Bei der Diskettenaufzeichnung existieren ja genau in diesem Sinne u. a. die beiden Codes FM und MFM. MFM ermöglicht zwar, weitaus mehr auf der gleichen Diskette unterzubringen, verlangt aber auch bessere Diskettenlaufwerke. Beim BASICODE hat man sich also für ein sicheres Verfahren zum Nachteil bezüglich der hohen Datenrate entschieden.

Einige Ergänzungen sollen die spezifischen Probleme noch vertiefen. Die Gleichstromfreiheit der beiden Signale gilt streng nur für ihre periodische Wiederkehr. Wirkliche Daten- oder Programm-Files wechseln aber in mehr oder weniger zufälliger Folge zwischen den beiden Signalen. Hierdurch entstehen zusätzliche Spektralkomponenten, welche vor allem zu tieferen Frequenzen beitragen. Deshalb auch die obige Forderung von 400 Hz statt der zunächst denkbaren 1200 Hz. In

extremen Fällen existieren natürlich noch tiefere Frequenzanteile. Sie zeigen sich aber praktisch vernachlässigbar. Dennoch bewirkt ihr Weglassen natürlich Störungen, die sich in einer additiven Amplitudenverschiebung bemerkbar machen. Ihr sollte durch die Empfängerschaltung, z. B. mittels eines Schmitt-Triggers, entgegengewirkt werden. So kann die additive Verschiebung des Mittelwertes weitgehend unwirksam bleiben.

Die Einengung der Bandbreite nach oben verformt vor allem die Rechtecksignale in die Richtung zu sinusähnlichen Signalen. Dadurch verschieben sich aber kaum die Nulldurchgänge. Sie bleiben weiterhin gut auswertbar. Selbst wenn dann die 1200 und 2400 Hz vom Kanal unterschiedlich stark (ohne Phasenfehler) gedämpft werden, tritt noch keine wesentliche Verschiebung auf. Die Originalwerte sind also immer noch recht zuverlässig zu detektieren. Die Verhältnisse werden erst dann erheblich ungünstiger, wenn Phasenfehler und Einschwingvorgänge auftreten. Wenn bei der Übertragung oder Speicherung gemäß dem BASICODE-Protokoll oder generell bei einem Kassetteninterface Phasenfehler auftreten, so ist es am besten, sich die Signale auf einem Oszillografen anzuschauen. Hierbei sollte unbedingt auf folgende Punkte geachtet werden:

- Die Nulldurchgänge sollten relativ steil verlaufen.
- Vor und hinter den Flanken dürfen keine höherfrequenten Einschwingvorgänge auftreten. (Wenn Sie auftreten, werden meist zu hohe Frequenzen zu stark übertragen. Es können auch starke Phasenfehler am oberen Frequenzende die Ursache sein.)
- Das Dach der angestrebten Rechteckschwingung darf keine zu große Schräge (10%) aufweisen. (Als Ursachen kommen hier in Betracht: zu wenig tiefe Frequenzen bzw. starke Phasenfehler in beiden tiefen Frequenzen.)
- Positive und negative Halbwellen sollten gleichartig aussehen (Phasenfehler).
- Möglichst gleiche Amplituden der beiden Frequenzen.

Die meisten dieser Fehler können durch Frequenzgangkorrektur in der Phase und/oder Amplitude ausgeglichen werden. Das BASICODE-Protokoll ist gegenüber diesen Fehlern relativ wenig störanfällig. Deshalb dürften sie eigentlich nur dann auftreten, wenn Ihr Kassettenrecorder unbrauchbar geworden ist. Nebenbei sei erwähnt, daß bei diesem Verfahren Recorder mit hifi-Qualität kaum besser, oft sogar ungünstiger abschneiden als einfache, billige Recorder. Recorder mit großer Bandbreite (hifi-Recorder, z. T. sogar Studiogeräte) sind speziell für akustische Anwendungen bis zu den höchsten Frequenzen vor allem im Amplitudengang oft auf Kosten des Phasenganges linearisiert. Dadurch treten nicht selten die oben genannten störenden Einschwingvorgänge auf. Dieses Problem hat uns auch zu Beginn bei den Rundfunkübertragungen Probleme bereitet. Bei Stereo-Recordern besteht außerdem noch die Möglichkeit, daß nicht beide Kanäle voll identisch sind. Bei der Monoschaltung können sich dann zwei etwas unterschiedliche Signale gegenseitig negativ beeinflussen. Deshalb sollte bei Ihnen möglichst nur ein Kanal und wenn möglich der rechte verwendet werden. Seine Spur liegt nämlich nicht am Außenrand des Magnetbandes. Ferner lassen sich einige Störungen durch angepaßte Aussteuerung (etwa Vollaussteuerung, u.a., weil dann weniger Rauschen auftritt und sich die Signale in Richtung zu Rechtecksignalen abflachen) senken. Ganz wichtig ist es, daß die Tonkopfstellung (Spaltlage) exakt 90° zur Bandlaufrichtung beträgt. Der Fachmann justiert dies mit einem speziellen Rauschband. Hin und wieder ist auch ein Säubern des Gerätes (insbesondere der Tonköpfe) notwendig. Abgelagerter Abrieb verändert den Frequenzgang deutlich. Aber auch gegen diese Mängel ist das BASICODE-Signal deutlich weniger anfällig als die anderen Verfahren.

4.2. Byte-Niveau

Werden 8 Bit zusammengefaßt, so entsteht ein Byte. Es stellt eine wesentliche Grundeinheit der Rechentechnik dar, welche eine umfangreichere Bedeutung besitzt. Genau diese 8 Bit bilden auch die nächste Einheit beim BASICODE-Protokoll. Sie werden eingerahmt von einem Start-Bit und zwei Stopp-Bit. So sind statt der eigentlichen 8 Daten-Bit insgesamt 11 Bit zu übertragen. Diese 11 Bit werden asynchron übertragen. Nach einer Pause zeigt das Start-Bit an, daß nun die 8 Daten-Bit folgen, und die zwei Stopp-Bit weisen darauf hin, daß die 8 Bit bereits übertragen wurden. So werden die 8 Daten-Bit mit weitaus größerer Sicherheit erkannt. Es wird also erneut mit großer Redundanz (Weitschweifigkeit, nützlicher Reserve) gearbeitet. Wenn nämlich einmal bei einer Übertragung oder Speicherung ein Fehler auftritt, so betrifft er meist nur ein oder wenige Bit. Durch die Start- und Stopp-Bit wird nach einem solchen Fehler aber mit großer Sicherheit wieder der Anfang des nächsten Byte erkannt. Ohne diese Redundanz wäre es sehr viel schwieriger, wieder in den richtigen "Rhythmus" für die folgenden Byte zu kommen. Infolge der hohen Redundanz ist es daher kaum mehr möglich, daß durch eine Störung alle darauf folgenden Daten unbrauchbar werden. Infolge dieser asynchronen Übertragung mit dem Start- und Stopp-Bit geht also maximal das gestörte Byte verloren, und da dies vielfach auch noch ein ASCII-Code ist, ist es meist relativ leicht zu rekonstruieren. Dies gilt ganz besonders für die BASICODE-Programme.

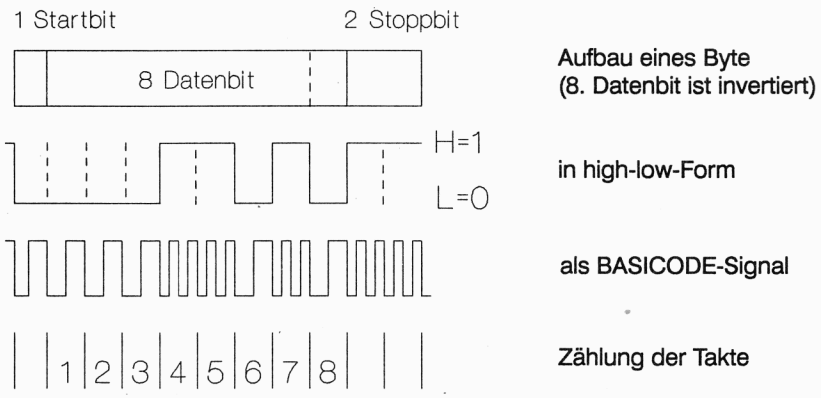


Abbildung 4.10.

Neben der bisher geschilderten Redundanz besitzt das BASICODE-Protokoll noch eine zusätzliche Sicherheitsmaßnahme. Es ist immer das 8. Daten-Bit negiert. Dieses Prinzip bewirkt bei ASCII-File indirekt ein zusätzliches, drittes Stopp-Bit. Die Datenübertragung beginnt stets mit Bit 0 und endet mit Bit 7. Im ASCII-Format ist nun aber immer Bit 7 low (0). Durch die Invertierung wird es high (1), wie die dann folgenden eigentlichen Stopp-Bit. Die Abbildung 4.8 zeigt beispielhaft ein Signal mit der Bit-Folge 0 0001 1010 11. Für die Umrechnung ist der gültige mittlere Teil rückwärts zu lesen und ergibt daher das Byte 0101 1000, also 58H, also Bit 7 negiert: 1101 1000 = D8H = 216D. Hier handelt es sich um kein ASCII-Zeichen!

Die im vorigen Abschnitt behandelte Detektion der BASICODE-Signale setzt meist an dieser Struktur an und kann häufig durch den folgenden formalen Ablauf beschrieben werden.

In der Ruhe sind ja 2400 Hz vorhanden. Bei jeder Flanke wird die Zeit bis zur nächsten Flanke gezählt. Sofern sie später als nach etwa 1/1800 Sekunden (1.5fache von 2400 Hz) erscheint, liegt das Start-Bit vor. Dann werden wieder 1/1800 Sekunden abgewartet und die Signallage getestet. Sie gibt an, ob das erste Daten-Bit eine 0 oder 1 ist. Nach jeweils 1/1200 Sekunden kann die Entscheidung

über die folgenden Bit getroffen werden. Nach dem Bit 7 des Byte wird wieder auf ein Start-Bit gewartet. Wegen der zwei (bzw. 3) Stopp-Bit spricht die Mehrzahl der guten Leseprogramme immer nach vier oder weniger 2400-Hz-Schwingungen bevorzugt auf ein Start-Bit an. Das eben geschilderte Prinzip der Decodierung von BASICODE-Signalen ist jedoch nur eine Variante von vielen möglichen. So kann auch fortlaufend die Zeit zwischen drei aufeinander folgenden Nulldurchgängen (also die Dauer einer ganzen Periode) gemessen werden.

Aus dieser Betrachtung ist weiter ersichtlich, daß die Frequenzen nicht exakt bei 1200 Hz und 2400 Hz zu liegen brauchen. Abweichungen um 10 % bereiten in der Regel keine Probleme. Es existieren aber auch "intelligente" Leseprogramme, die aus dem Anlaufton jene Parameter bestimmen, die das Magnetband mit dem Kassettenrecorder exakt ausgibt. Dann sind weitaus größere Geschwindigkeitsabweichungen zulässig.

4.3. Struktur von ASCII-Files und Programmen

Bereits oben wurde erwähnt, daß im BASICODE-Protokoll ASCII-Files und Programme nur aus einem Block bestehen. Die Möglichkeit hierfür liegt im eingeschränkten Bereich der zugelassenen Hex-Codes. Hierfür gilt:

- Alle ASCII-Zeichen von 32 bis 126 (20H bis 7EH)
- ENTER = 13 (0DH)

Für die wirklich gespeicherten bzw. übertragenen Daten ist zu beachten, daß dort noch zusätzlich Bit 7 negiert wird. Dadurch existieren dann genau die Werte 100 (0A0H) und 141 (8DH) bis 254 (0FEH). Andererseits fällt auf, daß nicht einmal 127 (7FH = Delete) und der Zeilenrücklauf 10 (0AH) zugelassen sind. Für die BASICODE-Programme dient das Zeichen 13 (0DH) zum Trennen der einzelnen Zeilen. Für eine direkte Druckerausgabe muß also nach 13 (CR) automatisch 10 (LF) angefügt werden.

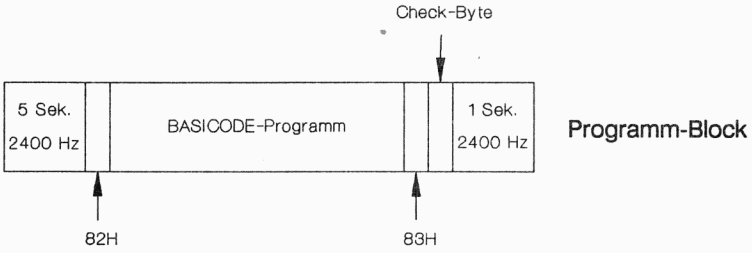


Abbildung 4.11.

Jedes ASCII-File bzw. BASICODE-Programm besteht nun aus einer Aufeinanderfolge der oben genannten HEX-Zeichen. Dabei ist die Länge dieses Files belanglos. Es wird stets als vollständige Einheit hintereinander übertragen. Aber so wie das Byte in einen Rahmen aus Start- und Stopp-Bit eingefügt ist, erfolgt nun ähnliches für das Programm.

Diesen Aufbau zeigt die Abbildung 4.9. und die Beschreibung lautet folgendermaßen:

- Beginn mit ca. 5 Sekunden 2400-Hz-Signal zum Einsynchronisieren des Empfängers. Dieser Teil wird oft Startton genannt.

- Ein Start-Byte von 82H, also 10000010 beendet den Startton und leitet den Beginn des eigentlichen ASCII-Files ein. Infolge der Invertierung von Bit 7 ist 82H eigentlich das übliche STX (start of text) 02H.
- Das ASCII-File bzw. das eigentliche Programm
- Den Abschluß bildet das Byte 83H, also 10000011, welches ohne Invertierung ETX = end of text 03H ist.
- Anschließend folgt ein berechnetes Sonderzeichen zur Fehlererkennung, nämlich ein Kontroll- oder Check-Byte. Lediglich hierzu folgt unten noch eine Erklärung.
- Den endgültigen Abschluß bildet ca. eine Sekunde 2400 Hz.

Das Checkbyte wird aus allen bis dahin übertragenen Byte durch bitweise Binär-Addition (XOR) berechnet. Für nur vier Byte demonstriert dies die folgende Tabelle:

HEX	Binär
82	1000 0010
C7	1100 0111
8D	1000 1101
83	1000 0011
Check-Byte =	0100 1011 = 4BH

In jeder Bitposition wird die Anzahl der auf 1 gesetzten Bit gezählt. Dabei werden alle Byte des ASCII-Files bzw. des Programms und auch das Anfangs-Byte 82H sowie das Ende-Byte 83H einbezogen. Wenn dabei eine geradzahlige Anzahl entsteht, so wird für die entsprechende Stelle das Check-Bit 0 gesetzt, andernfalls 1. Diese Berechnung erfolgt sowohl beim Senden (Aufzeichnen) als auch beim Empfangen (Wiedergeben). Beim Senden ist nach der Aufzeichnung dieses Check-Byte alles erledigt. Bei der Wiedergabe wird analog zum Senden ebenfalls das Check-Byte byteweise berechnet, und danach kann es mit dem gesendeten Check-Byte verglichen werden. Wenn beide Check-Byte übereinstimmen, liegt große Wahrscheinlichkeit dafür vor, daß zwischen gesendeten und empfangenen Daten kein Unterschied besteht, d. h., es liegt wahrscheinlich eine fehlerfreie Übertragung vor. Die Wahrscheinlichkeit für eine Fehlerfreiheit bei übereinstimmenden Check-Byte ist besser als $1:256^2 = 1:65536$. Wenn das gesendete und das beim Empfang berechnete Check-Byte nicht übereinstimmen, wird eine Fehleranzeige ausgelöst. Beim BASICODE-Programm oder ASCII-File ist aber auch dann oft noch, allerdings nur über den Kontext, eine Korrektur möglich.

Bezüglich des BASICODE-Programms sei an dieser Stelle noch ein Hinweis auf Möglichkeiten der KC-Rechner gemacht. Sie gestatten nämlich die Ausgabe ihrer Programme ebenfalls im ASCII-Format. Dies erfolgt mit dem Befehl LIST#1. Allerdings werden beim normalen Betrieb nach 0DH noch zehn 00-Byte angefügt. Dies kann jedoch mittels des direkten Befehls NULL 0 ausgeschaltet werden. *Bei einigen anderen Rechnern erfolgt eine ASCII-Programmausgabe mit dem Befehl* SAVE "Name",A. Es ist also ein Anfügen des Schalters A nach einem Komma notwendig.

4.4. Struktur von Datenfiles

Datenfiles werden oft auch als Binär-Files bezeichnet. Sie können im Gegensatz zu den ASCII-Files bzw. BASICODE-Programmen beliebige HEX-Werte im Bereich von 0 bis 255 (0H bis FFH) enthalten. Damit besteht nicht mehr die Möglichkeit, ihren Beginn und Ende durch ein spezielles Byte zu kennzeichnen. Eine Datenübertragung verlangt daher eine eindeutige Länge. Sie wurde im BASICODE-Protokoll auf 1024 Daten-Byte festgelegt. Dadurch entsteht eine Grundstruktur für einen Datenblock, wie es die Abbildung 4.10 zeigt:

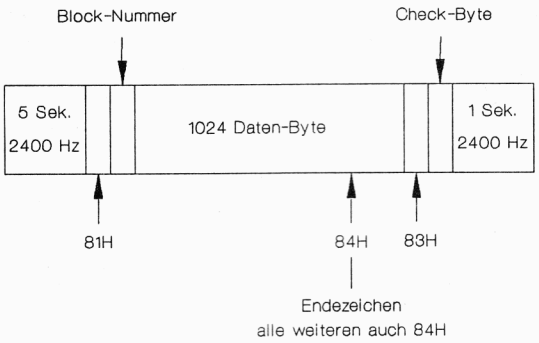


Abbildung 4.12. Datenblock-Struktur

- Zu Beginn werden 5 Sekunden 2400 Hz gesendet.
- Als Startkennzeichen folgt ein Byte mit dem Wert von 81H (01H = STH start of header). Es dient zur Unterscheidung gegenüber ASCII-File und BASICODE-Programmen, wo 82H verwendet wird .
- Außerdem wird vor dem Datenblock noch eine Blocknummer eingefügt. Sie lautet beim ersten Block 80H (0 wegen des invertierten Bit 7) und wird für jeden folgenden Block um eins erhöht. Der 5. Block trägt hier also die Nummer 84H.
- Schließlich folgen die 1024 Datenbyte (mit Bit 7 invertiert).
- Das Ende-Byte ist wieder 83H als ETX.
- Check-Byte wird über die vorangegangenen 1027 Byte gebildet.
- Den Abschluß bildet eine Sekunde 2400 Hz.

Aus derartigen Blocks wird nun ein komplettes Daten-File zusammengesetzt. Je nach der Größe des Daten-Files sind ein oder mehrere der o. g. Datenblocks notwendig. Dabei wird der letzte Block - welcher bei kurzen Files auch der einzige, erste sein kann - nicht immer genau die 1024 Daten-Byte ausfüllen können. Deshalb ist noch ein spezielles **Ende-Byte** definiert, nämlich 84H (04H = EOT = end of tape). Damit hier aber eine eindeutige Endekennzeichnung existiert, werden von hier ab alle restlichen Byte dieses Blockes auf 84H gesetzt.

Es ist bedeutsam, daß infolge des systematischen Aufbaus von Datenblocks das Bit 7 des Check-Byte immer 1 ist. Bei ASCII-Files kann es die Werte 0 oder 1 annehmen.

In BASICODE werden die Daten-Byte generell als String übergeben. Für die Ausgabe von HEX-Code 3 erfolgt die gemäß A$ = CHR$(3) durch das BASICODE-Programm.

4.5. Umgang mit Kassetten und Aufzeichnungen

Bei BASICODE erfolgt die Speicherung von Daten und Programmen auf Kassette mittels Kassettenrecordern. Beide sind ursprünglich für die analoge Speicherung von Ton, also nicht für digitale Daten und Programme entwickelt worden. Sie werden hier also "zweckentfremdet" verwendet. Dies bedingt einige Besonderheiten im Umgang mit den Kassetten und Geräten. Dabei kommt hinzu, daß die Erzeugung von Daten und Programmen meist mit großem Aufwand verbunden ist. Manchmal ist es fast nicht mehr möglich, sie erneut zu gewinnen. Daher muß dafür gesorgt werden, daß die Daten und Programme fehlerfrei auf die Kassette gelangen und dort sorgfältig vor Zerstörung und Verlust geschützt werden. Hieraus folgt, daß mit den Geräten und den Kassetten auf besondere Weise umzugehen ist. Da es sich bei den BASICODE fast immer um "private" Aufzeichnungen

handelt, brauchen andererseits hier nicht die Regelungen und Gesetze zum Datenschutz angewendet zu werden. Dennoch dürfte es sehr nützlich sein, Richtlinien zu besitzen. Ich werde hierzu vor allem von meinen Erfahrungen und Gepflogenheiten ausgehen. Hierbei ist deutlich zwischen dem Umgang mit den Kassetten und den Kassettenrecordern zu unterscheiden. Auf den Umgang mit Kassettenrecordern wird im nächsten Abschnitt eingegangen. Beim Umgang mit Kassetten sind zwei Fakten zu unterscheiden:

- Vermeidung von Einflüssen auf die Aufzeichnung
- Anlegen einer Sicherheitskopie.

Für Programme und Daten, die mit großem Aufwand erzeugt wurden und die längerfristig benötigt werden, ist es nützlich bis notwendig, eine **Sicherheitskopie** (back-up) zu besitzen. Dies bedeutet, daß derartige Aufzeichnungen auf eine zweite Kassette übertragen werden müssen. Diese Kassette ist besonders geschützt und relativ weit entfernt von den Arbeitskassetten an einem sicherem Ort aufzubewahren. Besonders geschützt heißt hier, sowohl gegen fremden, aber auch unbeabsichtigten Zugriff als zufällige Änderungen. Zusätzlich sollten hierbei natürlich alle Punkte beachtet werden, die im folgenden generell für die Sicherheit und Fehlerfreiheit einer Aufzeichnung behandelt werden.

Die **Fehlerfreiheit und Zuverlässigkeit** einer Aufzeichnung hängt von einer Vielzahl Einflüsse ab. Sie betreffen die Eigenschaften des Recorders und der Kassette. Hier wird zunächst vorausgesetzt, daß die Gerätetechnik in dem entsprechend guten Zustand ist. Dies ist dann etwa der Fall, wenn höchstens jede hundertste Speicherung mißlingt und die Aufzeichnungen auch von anderen BASICODE-Nutzern problemlos genutzt werden können. Es muß aber betont werden, daß eine immer absolut fehlerfreie Aufzeichnung nicht möglich ist. Fehlerfrei ist stets ein statistischer Wert, und selbst bei einer Fehlerrate vom 1:1 000 000 besteht durchaus die - wenn auch sehr unwahrscheinliche - Möglichkeit, daß gerade eine besonders wichtige Aufzeichnung unrettbar verlorengeht. Einen gewissen Schutz hiergegen bietet die Kontrolle der Aufzeichnung durch den meist vorhandenen Befehl VERIFY. Durch ihn wird aber lediglich sichergestellt, daß zu diesem Zeitpunkt die Aufzeichnung mit sehr großer Wahrscheinlichkeit (nicht aber absoluter Sicherheit) fehlerfrei ist. Ich halte es für günstiger - und habe es auch stets so praktiziert - ‚eine zweite Aufzeichnung unmittelbar an die erste anzuschließen. Dies dauert nicht länger als VERIFY und erhöht dazu die statistische Sicherheit. Hat meine Technik eine Fehlerrate von 1:1 000, so ist dann die Fehlerrate, daß beide Aufzeichnungen fehlerbehaftet sind, nur noch 1:1 000 000. Daß dann auch noch die gleichen Teile in beiden Aufzeichnungen betroffen sind, ist noch extrem viel geringer. Es müßte also eigentlich immer - selbst in den extremsten Fällen - möglich sein, bei dieser Technik aus beiden Aufzeichnungen die gesamte Aufzeichnung zu rekonstruieren. Ich kann Ihnen aber versichern, daß ich so etwas noch nicht einmal nötig hatte: Eine Aufzeichnung war immer fehlerfrei. Natürlich war meine Technik immer im recht guten Zustand.

Generell sind Kassetten vor allem vor Staub, Feuchtigkeit, Sonne, Hitze und Chemikalien zu schützen. Sie sollten daher immer in ihrer **Plastehülle** und gut gekennzeichnet aufbewahrt werden. Ein **Rückspulen** des Bandes sorgt für weitere Sicherheit, da dann nur das Vorspannband im vorderen Ausschnitt liegt und die gesamte Bandoberfläche im Wickel gut geschützt ist. Außerdem hat dies den Vorteil, daß die Zählwerksangaben beim Suchen einer bestimmten Aufzeichnung stimmen. Wenn das Band nicht zurückgespult ist, besteht die Gefahr, die Bandoberfläche im vorderen Ausschnitt der Kassette mit dem Finger oder anderen Gegenständen zu berühren. Ein Fingerabdruck auf der Bandoberfläche führt fast mit Sicherheit zu künftigen Fehlern.

Kassetten sollten höheren **Temperaturen** deshalb nicht ausgesetzt werden, weil dann Verformungen des Gehäuses der Kassette und des Bandmaterials auftreten können. Beides führt mit großer Wahrscheinlichkeit zu Fehlern.

Der mögliche Störeinfluß von **Magnetfelder** wird meist übertrieben. Das Magnetband hat eine relativ hohe Koerzitivfeldstärke, und daher ist nur mit relativ starken Feldern eine Veränderung der Aufzeichnung möglich. Natürlich sollten Sie nicht willkürlich mit einem starken Magnetfeld auf die Kassette einwirken. Im Haushalt und Büro gibt es jedoch eigentlich nur zwei gefährliche Einflüsse.

- Haftmagnete, wie sie bei einigen Türverschlüssen, Seifenhaltern usw. oder an Magnettafeln verwendet werden.
- Farbfernseher entwickeln für ca. eine Sekunde beim Einschalten zur Entmagnetisierung der Lochmaske ein starkes Magnetfeld.

In beiden Fällen genügt es jedoch, einen Abstand von mindestens 15 cm einzuhalten. Scheren, Büroklammern, Schraubenzieher, Netztrafos usw. verfügen über viel schwächere Magnetfelder. Sie können höchstens bei ganz direktem Kontakt mit dem Magnetband zu Störungen führen. Auch beim Transport von Kassetten kann kaum eine Störung auftreten. Nur wenn die Kassetten ganz ungünstig bei Elektrokarren liegen oder in eine Elektrolok mitgenommen und dort ebenfalls ungünstig gelagert werden, könnten eventuell Aufzeichnungen verändert werden. Gegenüber Magnetfeldern sind jedoch **Disketten**, magnetische Kontokarten usw. weitaus mehr gefährdet. Der geringe Abstand zur Magnetschicht ist hier entscheidend. Die Hülle einer Diskette ist z.B. für Magnetfelder vernachlässigbar dünn. Hier kann sogar eine Büroklammer gefährlich sein.

Schließlich ist es noch notwendig, einige Bemerkungen zu den verschiedenen **Kassettentypen** zu ergänzen. Einmal existieren Kassetten mit unterschiedlicher Aufnahmezeit. Für die Rechentechnik sollten keine Kassetten mit längerer Laufzeit als 60 Minuten (also keine C90 oder C120) verwendet werden. Sie enthalten sehr dünnes Magnetband. Bei den Computer-Anwendungen existieren viele Start-Stoppvorgänge und das Band wird sehr häufig vor- und zurückgespult. Hierdurch besteht die Gefahr einer mechanischen Überbeanspruchung des Bandes, was zur plastischen Verformung und damit zum Informationsverlust führen kann. Im Gegensatz zur Audiotechnik existieren für die Rechentechnik auch **Kurzkassetten** mit Aufnahmezeiten unter 60 Minuten, also z.B. C10, C15 und C20. Sie enthalten gleichdickes Band wie die C60-Kassetten. Ihr Vorteil liegt also nur in der besseren Nutzbarkeit. Ich benutzte insbesondere die C10 bei der Programmentwicklung. Hier entsteht der folgende Vorteil: Sie bieten die Möglichkeit zum schnellen Umspulen. Auf der A-Seite lege ich meine aktuelle Version zweimal ab. Dann wird die Kassette umgespult. Nach etwa einer halben Stunde Arbeit kommt auf die B-Seite zweimal die neue aktuelle Version. Die Kassette wird wieder sofort umgespult. So existieren zu jedem Zeitpunkt je 2mal die beiden letzten aktuellen Versionen.

Eine weitere wichtige Eigenschaft der Kassetten wird durch ihr **magnetisches Material** bestimmt. International werden vier **IEC-Klassen** unterschieden. Die Klasse I wird durch die einfachen Kassetten mit Gamma-Fe_2O_3 bestimmt. Sie genügt vollauf den Ansprüchen der rechentechnischen Aufzeichnung. Die Klasse II, die vor allem durch Chromdioxid-Bänder bestimmt wird, kann bei entsprechenden Kassettenrecordern durchaus Verwendung finden. Die höheren Klassen, u.a. Rein-Eisenbänder, sollten nicht nur des hohen Preises wegen vermieden werden. Sie liefern oft, wie noch unten gezeigt werden wird, einen zu guten Frequenzgang. Weitaus wichtiger als die IEC-Klasse sind jedoch andere Qualitätsmerkmale. Sie werden vor allem durch Dropout-Armut, Schnittkante des Bandes, Gleichmäßigkeit der Magnetschicht usw. bestimmt. Da diese Merkmale kaum mit einfachen Mitteln feststellbar sind, empfiehlt es sich, nur **Qualitätskassetten** zu verwenden. Sie sind in erster Linie am Markennamen zu erkennen.

4.6. Umgang mit Kassettenrecordern

Der Zustand (u.a. Pflege) eines Kassettenrecorders und seine Einstellungen (Aufzeichnungslautstärke, automatische oder manuelle Pegeleinstellung, Bandmaterial-Einstellung usw.) bestimmen ganz wesentlich die Fehlerfreiheit der Speicherung von Daten und Programmen. Über diese Qualität kann man sich einen ungefähren Überblick durch Registrieren der folgenden Fakten verschaffen. Wenn im obigen Sinne gute Kassetten verwendet werden, sollte höchstens jede zwanzigste Aufzeichnung Fehler erkennen lassen. Nach Möglichkeit sollte so gut wie nie ein Fehler auftreten, wenn die eigene Aufzeichnung wieder eingelesen wird.

Bei Fremdaufnahmen (sofern sie gut sind) sollten nur unwesentlich mehr Fehler auftreten. Das gleiche sollte für Ihre Aufzeichnungen bei anderen BASICODE-Nutzern gelten.

Wenn häufiger Fehler auftreten, sollten Sie etwas bezüglich Ihres Kassettenrecorders unternehmen. Hierbei können Sie etwa wie folgt vorgehen:

1. Bestimmen Sie die Fehlerhäufigkeit im Vergleich von eigenen und fremden Kassettenaufzeichnungen. Wenn vor allem fremde Aufzeichnungen nicht fehlerarm gelesen werden können, ist wahrscheinlich Ihre eigene Wiedergabetechnik nicht in Ordnung. Gehen Sie dann weiter zu bei Punkt 3.

2. Es sind also Fremdkassetten besser als die eigenen zu lesen. Deshalb muß ein Fehler bei Ihrer Aufzeichnungstechnik vorhanden sein. Dies kann vorrangig durch folgende Fehler bedingt sein:

2a. Ihre Aussteuerung ist nicht richtig. Versuchen Sie, den Aufzeichnungspegel zu verändern. Hierzu kann ein Lautstärkevergleich mit den Fremdkassetten nützlich sein. Beachten Sie auch den Unterschied von automatischer und manueller Aussteuerung. Meist funktioniert die automatische Aussteuerung recht gut.

2b. Sie verwenden eine falsche Kassettensorte, z.B. die Verwechslung von normalen Kassetten (IEC I) und Chromdioxidkassetten (IEC II) oder Kassetten minderer Qualität. Auch die Verwendung von Kassetten für hifi-Anwendungen (Metall- bzw. Reineisen-Kassetten IEC III oder IV) kann eine Ursache sein.

2c. Wenn Ihr Gerät Umschalter für Bandmaterial oder Rauschminderung usw. besitzt, kann auch die Vormagnetisierung oder Entzerrung (Zeitkonstante) bzw. Rauschminderung ungünstig bzw. falsch eingestellt sein.

2d. Falls Sie einen Stereo-Recorder besitzen, schalten Sie ihn auf Mono oder verwenden nur einen, möglichst den rechten Kanal. Einfache und preiswerte Monorecorder mit Zählwerk sind die günstigsten Geräte (siehe oben).

2e. Sind an Ihrem Gerät Höhen- und Tiefenregler vorhanden, so können diese eventuell falsch eingestellt sein. Dies ist jedoch nur bei speziellen Geräten möglich. Fast immer wirken diese Regler nicht auf den Diodenanschluß, der möglichst zu verwenden ist.

2f. In Ihrem Gerät liegt ein Fehler im Aufzeichnungskanal vor. Versuchen Sie es mit einem anderen Gerät bzw. bringen Sie Ihr Gerät in eine Werkstatt.

3. Ihr Gerät ist nicht Ordnung, falsch eingestellt oder gar defekt. Hierfür sind u.a. folgende Ursachen möglich:

3a. Der Tonkopf Ihres Gerätes ist falsch justiert. Sie können dies mit einem speziellen Rauschband oder einer industriellen Musikaufzeichnung nach der besten Höhenwiedergabe durch Verstellen der speziellen Schraube beheben. Achtung! Durch Neueinstellen können eventuell Aufzeichnungen, die zuvor lesbar waren, unlesbar werden (siehe unten).

3b. Der Tonkopf ist verschmutzt. Sie können ihn u. a. vorsichtig mit einem Wattestäbchen und Spiritus säubern.

3c. Der Tonkopf ist magnetisiert worden. Dies kann u. a. durch Kontakt mit magnetischen Geräten, z. B. durch einen Schraubendreher beim unvorsichtigen Einstellen nach 3a. erfolgen. Sie erkennen diesen Fakt daran, daß eine normale Tonwiedergabe beachtlich rauscht. Beheben kann dies nur ein Fachmann.

3d. Die Geschwindigkeit Ihres Recorders weicht zu stark von der Norm ab. Wenn Sie ein gutes Gehör haben, erkennen Sie dies an der unterschiedlichen Tonhöhe Ihrer und der fremden Aufzeichnungen. Auch die Messung der Laufzeit einer Kassette auf Ihrem und einem anderen Gerät kann hierüber Aufschluß geben. Der Regler für die Geschwindigkeitseinstellung im Gerät ist in der Regel zwar gut zugänglich, die Einstellung sollten Sie aber einem Fachmann überlassen. Beachten Sie auch hier - wie bei der Einstellung des Tonkopfes -, daß durch die Justage eventuell eigene Aufzeichnungen unlesbar werden können, die vorher lesbar waren.

Insgesamt zeigen also die Ausführungen - vor allem zur Tonkopf- und Geschwindigkeits-Justage -, daß eine **regelmäßige** Kontrolle Ihres Recorders angebracht ist. Sie sollten sich also in gewissen Abständen von der Austauschbarkeit Ihrer Kassetten und der Einlesbarkeit von fremden Kassetten überzeugen. Auch eine regelmäßige Kontrolle der Spaltlage des Tonkopfes mit einem Industrie- oder Meßband ist nützlich.

Schließlich noch einige Bemerkungen zu **Computer-Viren**. Ein grundsätzlicher Vorteil bei BASICODE ist, daß es bei Kassettenaufzeichnungen kaum möglich ist, daß Computer-Viren wirksam werden. Dennoch können sie hierüber mittelbar auf Rechner mit Disketten und Festplatten eingeschleust werden. Hierauf ist eventuell zu achten, wenn "unseriöse" BASICODE-Programme auf solche Rechner übernommen werden. Ein gewisser indirekter Schutz besteht aber bereits darin, daß sich der Autor eines BASICODE-Programms im Programm selbst ausweisen muß.

5. Beschreibung der Bascoder

In den folgenden Abschnitten werden die Bascoder für die verschiedenen Rechner ausführlich beschrieben. Dabei wird zunächst mit einer möglichst einfachen Einführung begonnen. Hierdurch soll erreicht werden, daß es ohne Vorkenntnisse möglich ist, BASICODE zu benutzen. Erst dann erfolgt eine detailliertere Beschreibung der spezifischen Eigenschaften des jeweiligen Bacoders. Für die unterschiedlichen Bascoder zeichnen jeweils spezialisierte Autoren verantwortlich. Sie werden in diesem Kapitel nicht besonders erwähnt, sind aus Kapitel 9 zu entnehmen. Viele Bascoder wurden von holländischen Spezialisten entwickelt und von DDR-Spezialisten in die deutschsprachige Form übertragen. Deshalb sind im Kapitel 9. auch die holländischen Spezialisten vorgestellt.

5.1. Bascoder für den KC85/3

Einfache Anleitung

Beim KC85/3 erscheint nach dem Einschalten das bekannte Menü. Der Cursor wird dann auf die Zeile

```
%BASIC
```

gesetzt und die rechts unten auf der Tastatur liegende Taste (im folgenden ENTER genannt) betätigt. Daraufhin erscheint die Abfrage:

```
HC-BASIC
MEMORY  END ? :
```

Bei Bestätigung durch ENTER erfolgt - sofern keine Speichererweiterung vorliegt - die ergänzende Ausschrift:

```
15Ø86 BYTES FREE
OK
>
```

Bereiten Sie nun das Einlesen des Bascoders von Kassette vor. Alle Bascoder-Versionen des KC85/3 tragen den Namen "BAC853". Sie geben also ein:

```
>CLOAD"BAC853"
```

Diese Eingabe ist zunächst noch nicht mit ENTER zu bestätigen. Die Kassette mit dem Bascoder wird in den Kassettenrecorder eingelegt und durch schnellen Vor- bzw. Rücklauf an die Startstelle (Zählwerk beachten) positioniert. Schalten Sie den Recorder auf Wiedergabe. Beim Ertönen des Starttones wird an der Rechnertastatur die obige Eingabe mit ENTER quittiert. Jetzt kann man das blockweise Einlesen des Bascoders entsprechend den auf dem Bildschirm angezeigten Blocknummern verfolgen. Nach ca. 2 Minuten erscheint:

```
OK
 >
```

Der Bascoder ist damit ähnlich wie ein normales BASIC-Programm geladen. Aber noch befindet sich der Rechner nicht im BASICODE-Mode. Hierzu führen Sie den Befehl:

```
>RUN
```

aus.

Auf dem Bildschirm entsteht das Startbild mit Hinweisen auf die verwendete Version, KC-Typ, Autoren des Bascoders und das BASICODE-Menü.

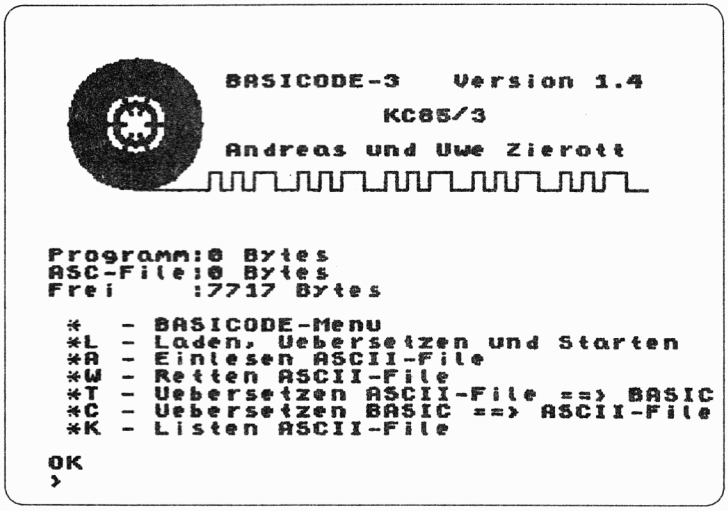


Abbildung 5.13. Startbild KC 85/3

Der Bildschirm zeigt also:

```
Programm:Ø Bytes        (Ø Bytes durch BASIC-Programm belegt)
ASC-File:Ø Bytes        (Ø Bytes durch ASCII/BASICODE belegt)
Frei     :x Bytes       (x Bytes im Arbeitsspeicher frei)
```

Diese Angaben stellen den jeweiligen Status dar. Die Anzahl der freien Bytes klammert den reservierten Stringspeicher aus und ist auch von der verwendeten Bascoder-Version und den Speichererweiterungen abhängig. Der Bascoder ist jetzt zur Funktionseingabe bereit und bietet neue Kommandos an:

```
  *  - BASICODE - Menu
  *L - Laden, Uebersetzen und Starten
  *A - Einlesen ASCII-File
  *W - Retten ASCII-File
  *T - Uebersetzen ASCII-File in BASIC
  *C - Uebersetzen BASIC in ASCII-File
  *K - Listen ASCII-File
OK
>
```

Ab hier können Sie ein BASICODE-Programm über die Tastatur Ihres Rechners eingeben oder mit dem Kommando *L ein BASICODE-Programm einlesen. Hierzu fordert Sie nach Betätigung der ENTER-Taste die Ausschrift:

```
Bitte Band starten
```

auf. Beim Einlesen können Sie den "Programmtext" verfolgen. Nachdem das Programm eingelesen ist, erfolgt automatisch die Übersetzung des BASICODE-Programms in die für den KC85/3 lesbare Form. Im Anschluß daran startet das Programm selbständig. Auf mögliche Fehler bei diesen Vorgängen wird weiter unten eingegangen. Ebenso werden im folgenden die einzelnen Funktionen detaillierter beschrieben.

Achten Sie bei der Kommandoeingabe bitte auch darauf, in welchem Status sich der Rechner befindet. Jede BASICODE-Menüausgabe teilt Ihnen mit, ob und in welcher Größe ein

- Programm (BASIC-Programm mit KC-Token)
  und/oder ein
- ASCII-File (BASICODE-Programmtext oder andere ASCII-Datei)

den Speicher Ihres KC's belegt. Die Anzeige des freien Speicherplatzes bezieht sich auf den Programmspeicher und berücksichtigt die Größe der ASCII-Files nicht.

Es kommt zu Fehlerausschriften, wenn Sie Funktionen aufrufen, die sich auf einen nicht vorhandenen Status beziehen. So kann ein ASCII-File nicht mit RUN gestartet werden, und ein KC-Programm läßt sich ohne Übersetzung (*C) nicht mit der Option *W auf Kassette retten.

Hinweise für den Programmierer

Der Bascoder des KC85/3 ist kein neuer BASIC-Dialekt! Den zugelassenen Befehlsvorrat von BASICODE kennen Sie schließlich schon von Ihrem KC-BASIC. Um ein echter BASICODE-Programmierer zu sein, sind jedoch die Regeln strengstens zu beachten. So sind z. B. bei den logischen Operatoren die zugehörigen Ausdrücke einzuklammern, RESTORE existiert nur ohne Zeilennummer und NEXT verlangt die Laufvariable. Ihr Bascoder kann nicht alle Regelverstöße wettmachen, wird Sie aber in der Option *C ggf. daran erinnern.

Viele der nicht zugelassenen BASIC-Befehle wie LINE, LOCATE oder SOUND werden durch die "versteckten" Subroutinen realisiert. Dort finden Sie auch Funktionen, die der KC85/3 bislang weder in BASIC noch mit den Routinen seines Betriebssystems zu bieten hatte. Dazu gehören das Aus- bzw. Einschalten der STOP / BRK-Taste, Textausgabe im Grafik-Mode und das Löschen von Linien.

Betrachten Sie den Bascoder deshalb einfach als Erweiterung des Betriebssystems.

Zum Verständnis des Bascoders gehört der Umgang mit den Ladebefehlen für die Kassettenarbeit. Ist BASICODE initialisiert, haben Sie neue Ladebefehle (*L, *A, *W), um mit dem für alle Bascoder verbindlichen Kassetteninterface arbeiten zu können. Nur auf diese Weise ist der Informationsaustausch zwischen den verschiedensten Computertypen gesichert. Die Ladebefehle des KC-BASIC haben jedoch nicht an Bedeutung verloren.

```
>CLOAD"NAME"
```

können Sie nutzen, um ein beliebiges KC-BASIC-Programm in den Speicher zu laden.

Speziell kann das ein in KC-BASIC transformiertes BASICODE-Programm sein, das zuvor mit CSAVE"NAME" auf Kassette gesichert wurde. Diese Verfahrensweise ist besonders für die Programmentwicklung anzuraten.

Halten Sie ein BASIC-Programm für geeignet, es den Regeln von BASICODE zu unterwerfen, es also anderen Computertypen zugänglich zu machen, können Sie es zunächst ohne spezielle Programmänderungen dem Bascoder übergeben. Ob der Programmspeicher frei ist, erkennen Sie im Menü. Ist der Wert für "Programm:" > 0 Bytes (Anzeige bei LIST), kann mit

```
>NEW
```

der Programmspeicher gelöscht werden. Ein sofortiger Start nach CLOAD führt mit ziemlicher Sicherheit zu einem Fehler. Dafür gibt es zumindest drei Gründe:

- Der Bascoder benötigt eine Zeile 1000 mit "GOTO 20".
- Der eigentliche Programmstart ist immer in Zeile 1010.
- Die Zeilennummern von 0-950 sind blockiert, sie werden nach RUN vom Bascoder eingeblendet und sind deshalb im Anwenderprogramm wirkungslos.

Hier hilft in jedem Fall das Kommando RENUMBER. Wird es wie üblich im Direkt-Mode verwendet, kann es die Subroutinen des Bascoders nicht beeinflussen.

Als Beispiel sei angenommen, das anzupassende Programm beginne mit Zeile 10 und ende bei Zeile 5830. Um eine geeignete Test-Struktur zu erreichen, tippen Sie eine Zeile 9 ein.

```
>9 A=1ØØ:GOTO2Ø:REM ** Programmname **
```

Die Zeile 9 soll unsere Zeile 1000 werden. Deshalb jetzt:

```
>RENUMBER 9,583Ø,1ØØØ,1Ø
```

Damit haben Sie eine brauchbare Ausgangsbasis auf dem Weg zu einem BASICODE-Programm. Auch später können Sie das Programm oder einzelne Teile von ihm mit RENUMBER neu ordnen und mit einem einfachen Trick sogar die Reihenfolge der Befehlszeilen ändern (geforderte Programmstrukturen beachten). Dazu müssen jedoch zunächst schon einmal alle nicht erlaubten BASIC-Befehle durch die entsprechenden Subroutinen ersetzt bzw. vollständig entfernt worden sein. Nehmen wir an, das vorliegende Programm hat seine DATA-Zeilen nicht zusammenhängend und im Programm verstreut. Durch:

```
RENUMBER (ab alte Zeile)(,bis alte Zeile)(,ab neue Zeile),10
```

werden alle Gruppen von DATA-Zeilen nacheinander auf den Bereich ab Zeile 25000 numeriert. Das LIST-Kommando zeigt zwar die erfolgte Veränderung in den Zeilennummern, ihre Plazierung im Programm hat sich jedoch nicht verändert.

Das Programm darf jetzt keinesfalls gestartet oder editiert werden, sonst möglicherweise Systemabsturz !

Um die Zeilen in die gewünschte Reihenfolge zu bringen, sind lediglich zwei BASICODE-Kommandos nacheinander auszuführen:

```
*C - Übersetzen BASIC in ASCII-File
*T - Übersetzen ASCII-File in BASIC
```

Noch einen letzten Hinweis zu RENUMBER. Liegt ein Programm gemäß dem BASICODE-Protokoll vor, sind also bereits Subroutinen der Bascoders eingebunden, sollte auch nicht zum Zwecke der Manipulation unter 1000 numeriert werden.

Es kann viele Gründe geben, BASICODE-Programme als KC-Tokenfile abzuspeichern. Beim KC85/3 mit D004-Erweiterung kommt noch hinzu, daß der Bascoder auch die Diskettenarbeit durch Verwendung der Umlenkroutine BASEX unterstützt.

Befindet sich der KC im BASICODE-Zustand, kann mit dem Befehl

```
>CSAVE "NAME"
```

das aktuelle Programm-File auf Kassette gespeichert werden. Dabei bleiben die Subroutinen des Bascoders ausgeblendet.

Achtung! Steht CSAVE nicht am Zeilenanfang, wird das betreffende Programm im Verbund mit dem Bascoder (ohne V24I) gerettet.

Nur für KC-Freaks

Erste Reaktionen auf die Entwicklung der KC-Bascoder zeigen, das nicht wenige Software-Spezialisten bemüht sind, hinter die "Geheimnisse" der Subroutinen zu kommen oder diese gar zu verändern. Tatsächlich ist kein Programm so perfekt, daß es nicht wesentlich verbessert werden könnte. Wir sind an ihren Gedanken und Vorschlägen immer interessiert, weniger dagegen an Rückfragen zu den nicht autorisierten Bascoder-Fassungen. Tatsächlich haben insbesonders die Bascoder der KC's aus Mühlhausen durch die umständliche IRM-Steuerung einen recht komplizierten Aufbau. Von Veränderungen ist deshalb abzuraten, zumal Sie davon ausgehen dürfen, niemals die aktuellste Version zu besitzen.

Ein durchaus "legales Betätigungsfeld" bietet der Bascoder bei Nutzung seiner Subroutinen für maschinennahe, typabhängige Programmierung. Sollen derartige Programme unabhängig von der verwendeten Version sein, sind die Subroutinen wie gewohnt mit GOSUB aufzurufen.

```
>GOSUB 1ØØ : LIST
```

zeigt die versteckten Subroutinen, an denen nicht editiert werden darf.
Die Adressen (CALL*XXXX) unterscheiden sich zwischen den KC-Typen und allen Bascoder-Versionen. Werden sie für besondere Anwendungen trotzdem direkt aufgerufen, könnte mit:

```
>:CSAVE"NAME"          (CSAVE ist nicht am Zeilenanfang)
```

das Programm im Verbund mit dem Bascoder als eine Art Laufzeitmodul auf Kassette gespeichert werden. Solcherart Programme haben mit BASICODE nichts als das "erweiterte" Betriebssystem gemeinsam. Da diese Programme das "Laufzeitmodul" schon enthalten, ist "BAC853" nach dem BASIC-Kaltstart nicht zu laden. Statt dessen ergibt sich folgender Ablauf bis zum Programmstart:

```
>CLOAD"NAME"          (spezielles Programm im Bascoder-Verbund)
>CALL*4ØD             (Initialisierung Bascoder / Startbild)
>RUN                  (Programmstart)
```

Und nun noch ein letzter Hinweis, der nur bedingt etwas mit den eigentlichen Bascoder-Funktionen zu tun hat. Im BASICODE-Menü und in der Beschreibung taucht mehrfach der Begriff ASCII-File statt BASICODE-Programm auf. Ein BASICODE-Programm ist, soweit es mit *C übersetzt oder mit *A geladen wurde, immer ein ASCII-File. Hingegen muß ein ASCII-File nicht zwangsläufig auch ein BASICODE-Programm sein. Denkbar ist dabei nicht nur die Vermittlung von Quelltexten verschiedener Programmiersprachen, sondern auch "elektronische Computermagazine" oder andere Texte.

Befehlsübersicht

Die Auswahl der Funktionen des Bascoders erfolgt durch die Eingabe eines Sternchens "*", dem ein Buchstabe und ENTER folgt. Sie können aber auch die Funktionstasten (F1-F6) nutzen. Über ihre Zweitbelegungen, sie werden hier als F7-FC bezeichnet, erreichen Sie sechs weitere Kommandos.

Als BASICODE-Kommandos (*) sind sowohl Groß- als auch Kleinbuchstaben zugelassen.

Eingabe	Funktion
*	BASICODE - Menü Es erscheint das Startbild mit allen Funktionen.
*L (F1)	Laden, Übersetzen und Starten So wird ein BASICODE-Programm von Kassette eingelesen, automatisch in das KC-BASIC übersetzt und gestartet.
*A (F2)	Einlesen ASCII-File Soll ein Programm oder ein ASCII-File im BASICODE-Format von Kassette in den Speicher geladen werden, wählen Sie diese Funktion. Da nicht übersetzt und gestartet wird, ist sie auch geeignet, Kopien im BASICODE-Format zu erstellen (*W).
*W (F3)	Retten ASCII-File Mit dieser Funktion retten Sie ein Programm oder ein anderes ASCII-File im BASICODE-Format auf Kassette.
*T (F4)	Übersetzen ASCII-File in BASIC Ein im Speicher stehendes Programm im BASICODE-Format wird durch diese Funktion in das KC-BASIC transformiert.
*C (F5)	Übersetzen BASIC in ASCII-File Ein im Speicher stehendes BASIC-Programm (KC-Format) wird in ein ASCII-File (BASICODE-Format) transformiert.
*K (F6)	Listen ASCII-File Ein im Speicher stehendes Programm (BASICODE) oder anderes ASCII-File wird auf dem Bildschirm zur Anzeige gebracht.
(F7)	RUN - Start eines BASIC-Programms
(F8)	CONT - Fortsetzung nach BRK
(F9)	LIST - Listen im KC-Format
(FA)	%REBASIC : CALL*40D - BASICODE-Restart nach RESET
(FB)	CLOAD" - Lade BASIC im KC-Format
(FC)	CSAVE" - Rette BASIC im KC-Format

Jedes "* - Kommando" bewirkt vor seiner Ausführung die Löschung des Bildschirms.

* BASICODE - Menü

Geben Sie nur ein Sternchen ein oder ist das ihm folgende Kommando nicht in der Liste vorhanden, erscheinen das Startbild, die aktuellen Werte der Speicheraufteilung und das oben gezeigte Menü der möglichen Funktionen auf dem Bildschirm.

*L Laden, Übersetzen und Starten (F1)

Die Option *L nutzen Sie, um mit einem Programm, das im BASICODE-Format auf Kassette steht, zu arbeiten.

Zunächst werden Sie aufgefordert, den Recorder zu starten. Erkennt der Bascoder den Vorton, werden von ihm die empfangenen Schwingungen ausgewertet. Die Lese-Routine berücksichtigt die erkannte Frequenz. Kann das Startbyte nicht eindeutig ausgewertet werden, erscheint die Ausschrift:

```
kein BASICODE-Programm !
```

Das Einlesen des ASCII-Files (BASICODE) kann mit geübtem Auge auf der untersten Bildschirmzeile verfolgt werden. Treten offensichtliche Lesefehler auf und Sie wollen das Einlesen abbrechen, so drücken Sie irgendeine Taste.

Ist das Einlesen beendet oder unterbrochen worden, erhalten Sie eine Ausschrift zum Stoppen des Recorders. Ist der verfügbare Speicherplatz zur Ablage des Programms nicht ausreichend, werden Sie mit der Meldung:

```
Speichergrenze erreicht!
```

gewarnt.

Hat der Bascoder alle Zeichen fehlerfrei lesen können, schließen sich automatisch die Transformation in das KC-BASIC (Option *T) und der Programmstart (RUN) an. Anderenfalls erhalten Sie den Hinweis:

```
"Lesefehler erkannt !"
```

In diesem Fall können Sie eventuell dennoch das ASCII-File mit der *T-Option zu einem editierbaren BASIC-Programm im (KC-Format) machen.

***A Einlesen ASCII-File (F2)**

Diese Option wird verwendet, um ein ASCII-File (im BASICODE-Format) in den Speicher einzulesen. Im Gegensatz zu *L muß das gelesene File nicht zwangsläufig ein echtes BASICODE-Programm sein. Unzulässig sind jedoch BASICODE-Bestände. Der Anwender übernimmt die Steuerung und kann entscheiden, ob ins KC-Format übersetzt mit *W eine Kopie herzustellen ist oder z. B. mit *K Text zur Anzeige gebracht werden soll. Ansonsten gelten die Erläuterungen zu *L.

***W Retten ASCII-File (F3)**

Zum Kopieren eines ASCII-Files (BASICODE) oder zur Aufzeichnung eines eigenen Programms im BASICODE-Format müssen Sie die *W- Option aufrufen. Zuvor sollte die Kassette positioniert, der Recorder auf "Aufnahme" gestellt und gestartet werden. Ist im Menü das ASCII-File mit 0 Bytes ausgewiesen, erscheint die Warnung:

```
kein ASCII-File im Speicher !
```

Das Auslagern auf Kassette kann in der Statuszeile verfolgt werden. Wollen Sie die Funktion vorzeitig beenden, betätigen Sie irgendeine Taste.

***T Übersetzen ASCII-File in BASIC (F4)**

Das Übersetzen eines BASICODE-Programms (ASCII-File) in das KC-Format führt der Bascoder nach dem fehlerlosen Einladen mit der Option *L automatisch aus. Durch *T kann unabhängig vom Ladevorgang gearbeitet werden. Das ASCII-File bleibt dabei nicht erhalten. Die übersetzten BASIC-Zeilen sind in der Statuszeile sichtbar.

Da der BASIC-Interpreter den Befehl LOG nicht kennt, hat der Bascoder die zusätzliche Aufgabe, bei der Wandlung des ASCII-Files (BASICODE-Programm) in KC-BASIC LOG durch LN zu ersetzen.

***C Übersetzen BASIC in ASCII-File (F5)**

Liegt ein Programm im KC-Format vor, wird es zunächst auf unzulässige Befehle durchsucht. Damit soll verhindert werden, daß BASIC-Anweisungen, die dem KC-Interpreter bekannt, aber nicht im Befehlssatz von BASICODE-3 vereinbart sind, in das ASCII-File gelangen.

Gleichzeitig werden gemäß den Regeln, wie ein ASCII-File auszusehen hat, Leerzeichen (SPACE) gesetzt bzw. entfernt. Logische Operatoren werden auf Klammerung der Ausdrücke getestet. Weitere Kontrollen beziehen sich auf RESTORE (ohne Zeilennummer) und NEXT (mit Laufvariable).

Findet der Bascoder einen Fehler, erhalten Sie die Meldung:

```
unerlaubter BASICODE-Befehl !
```

Oder etwas harmloser:

```
Regelverstoss !
```

Die betreffende BASIC-Zeile wird angezeigt und zur Editierung angeboten. Sie können nun den falschen Befehl verbessern. Nach ENTER befinden Sie sich im Direkt-Mode. Mit *C können Sie einen erneuten Übersetzungsversuch starten.

Nicht jeder Verstoß gegen das BASICODE-Protokoll läßt sich auf diese Weise aufspüren. Wurden Editierungen vorgenommen, macht sich ggf. ein erneuter Programmtest notwendig.

Bei Fehlerfreiheit im ersten Teil beginnt automatisch der zweite Teil - die eigentliche Übersetzung in ein ASCII-File. Dieser Vorgang ist wieder in der Statuszeile zu verfolgen. Dabei wird das BASIC-Programm an das Speicherende geschoben und das ASCII-File Zeile für Zeile an dessen ursprüngliche Stelle gesetzt. Textzeilen, die länger als die zulässigen 60 Zeichen sind, werden als "zu lang" auf dem Bildschirm notiert, der Befehl LN gegen LOG ausgetauscht.

Sind keine Fehlerausschriften zu verzeichnen und arbeitete das Programm im KC-Format fehlerfrei, haben Sie jetzt ein BASICODE-Programm (ASCII), das Sie mit *W auf Kassette schreiben können. Dieses Programm-File kann nun auf jedem Computertyp eingelesen werden, für den es einen Bascoder gibt.

Ist das Programm noch nicht fehlerfrei (Zeilen zu lang), ist es wichtig, Sie transformieren mit *T in das KC-BASIC zurück und führen entsprechende Änderungen durch. Sie gefährden sonst den Programmaustausch zwischen den Computertypen, das wichtigste Anliegen von BASICODE.

*K Listen ASCII-File (F6)

Mit der LIST-Taste oder dem Kommando LIST kann ein Programm im KC-Format, nicht aber ein ASCII-File angezeigt werden. Wollen Sie dieses jedoch sehen (z. B. unmittelbar nach fehlerhaftem Einlesen), so benutzen Sie die Option *K. Fehlerhaft gelesene ASCII-Zeichen, die dadurch einen unzulässigen Charakter erhalten haben (Steuerzeichen), werden als "#" dargestellt. Ab Version 2.0 soll Option *K einen Texteditor aufrufen. Mit ihm werden sowohl Korrekturen im ASCII-File vor der Übersetzung (*T) als auch Erstellung und Editierung BASIC-unabhängiger Files zur Abspeicherung im BASICODE-Format (*W) = möglich sein. Es existiert kein gesonderter Befehl zum Löschen eines ASCII-Files. Dies erfolgt automatisch bei jeder Art der Programmeingabe.

Arbeiten mit BASICODE-3-Beständen

Die Bestandsroutinen arbeiten für NF = 0 und NF = 1 mit BASICODE-Beständen gemäß den Festlegungen. Für die Arbeit im KC-Format (NF = 2;NF = 3) werden spezielle Routinen bereitgestellt. Bei Beachtung der zulässigen Zeichen sind u. a. auch Speicherabzüge (z. B. mit Textprogrammen erstellte Daten), insbesondere aber auch KC-Programmlistings und Stringfelder über ein BASICODE-Hilfsprogramm in einen echten BASICODE-Bestand wandelbar.
KC-seitig erhalten Sie das geforderte ASCII-Format durch:

```
PRINT#1 Daten
```

Oder für BASIC-Programmlistings:

```
LIST#1"Name"
```

Beachten Sie die Hinweise in Ihrem BASIC-Handbuch. In jedem Fall sollte die Anweisung NULL 0 gegeben werden.

Haben Sie einen BASICODE-Bestand, der mit NF = 3 (KC-Format) auf Kassette gerettet wurde, so kann er ebenfalls außerhalb BASICODE dem KC mit folgenden Befehlen übergeben werden:

```
INPUT#1 Daten
```

Oder für BASIC-Programmlistings:

```
LOAD#1"Name"
```

Auch hier sollte dann die NULL 0 - Anweisung zur Synchronisation gegeben werden. Der Anwender hat zusätzlich zu beachten, daß die mit NF = 3 gespeicherten Daten, genau wie die KC-Kanalausgaben, beim Einlesen im KC-Mode nach dem Endekennzeichen (03H) in den Direkt-Mode des Computers führen. Gegebenenfalls sind eigene Endekennzeichen zu definieren.

Durch wechselseitige Verwendung der BASICODE-Bestandsroutinen stehen dem KC-Nutzer also zusätzliche Verarbeitungsformen offen. Dabei kann der Bascoder entweder Hilfsmittel oder Betriebssystem sein.

Das Laden und Retten von Beständen auf Diskette (NF = 4 bis NF = 7) ist in der vorliegenden Version nicht vorgesehen. Hinweise zum Start bzw. Stop des Recorders werden gegeben. Zwischen den Blöcken braucht der Recorder nicht abgeschaltet zu werden, da das Auffüllen des Textpuffers recht schnell vor sich geht.

Sollen BASICODE-Bestände gelesen werden, prüft das Programm, ob es sich auch wirklich um eine Bestandsdatei handelt. Der Hinweis:

```
kein BASICODE-Bestand!
```

zeigt an, daß der Bascoder keine Bestände vorfindet. Wird ein Bestandsblock fehlerhaft gelesen, folgt die Ausschrift:

```
Lesefehler! zurueck? J/N
```

Kann Ihr Computer den fehlerhaften Block wiederholt nicht lesen, geben Sie "N" ein. Die Daten werden dann trotz Fehler übernommen. Ein gutes BASICODE-Programm wertet diese dann aus.

BASICODE-Restart

Verlassen Sie gewollt BASICODE, etwa um die V24-Schnittstelle zu initialisieren, ist dies mit den gewohnten Kommandos zu erreichen:

```
>BYE            BASICODE in CAOS
%REBASIC        CAOS in BASICODE
```

Der Systemabsturz gehört zu jedem guten KC! Auch BASICODE ändert daran nicht viel. Einige CAOS-Systemfehler werden zwar ausgemerzt, andere mit Sicherheit produziert. Wenn Sie Glück im Unglück haben, verrät Ihnen der Bascoder noch:

```
!Systemabsturz!
```

und Sie haben Ihr Programm nicht verloren. Häufiges Auftreten dieser Erscheinung weist auf einen Gerätefehler oder eine verfälschte Bascoder-Version hin.

War es unumgänglich, den BASICODE-Zustand zu verlassen (RESET), kann dieser u. U. erneut hergestellt werden, ohne den Bascoder nochmals einzuladen. Betätigen Sie dazu die Funktionstaste FA (SHIFT + F4). Bringt das noch keinen Erfolg, ist zunächst

```
%REBASIC
```

aufzurufen. Erscheint die Bestätigung mit

```
OK
>
```

geben Sie bitte den Befehl

```
>CALL*4ØD
```

ein und lassen Sie ihn durch ENTER ausführen. Somit werden, soweit das noch möglich ist, die Systemzeiger auf BASICODE gesetzt. Sie erkennen den ordnungsgemäßen RESTART am Startbild.

Besonderheiten des KC85/3

Die Bedienoberfläche des Bascoders wurde für alle KC-Typen einheitlich gestaltet. Die Gleichheit der Kommandos und Ähnlichkeiten in ihrer Eingabe zwischen KC und ZX-Spectrum sind ebenfalls gewollt. Wir hoffen, daß diese Tatsache zu einer schnellen Verbreitung und Nutzung von BASICODE beiträgt.

Der Textschirm des KC85/3 besteht aus 32 Zeilen mit je 40 Zeichen. Die Vollgrafik (256*320) nutzt der Bascoder mit 240*320 Pixel aus. Der Bascoder des KC85/3 besitzt neue Routinen [15] für seine Linienbefehle und entwickelt dabei eine höhere Arbeitsgeschwindigkeit als der KC 85/4.

Der ASCII-Zeichensatz des KC85/3 weicht vom Standard (US-ASCII) ab. Die Codes 91-93 und 96 sind über die Tastatur zu erreichen, geben aber andere Zeichen auf dem Bildschirm aus. Auch die Codes 123-126 nutzt der KC zur Darstellung anderer Zeichen (ä,ö,ü,ß). Hierfür sind jedoch keine Eingabetasten vorgesehen. Der Bascoder berücksichtigt diesen Umstand nicht weiter, da eine große Anzahl von Computern in ähnlicher Lage sind und somit diese Codes besser nicht verwendet werden sollten. Ist innerhalb von BASICODE-Programmen ein Zeichen zu löschen, so ist dazu die Taste CLR vorgesehen. Sie gibt den Code 127 ab. Im EDIT-Modus hat sie ihre alte Bedeutung verloren.

Die Forderungen der Subroutine 650 wurden realisiert. Schrift kann auch unabhängig von den Textzeilen grafisch geschrieben und wieder gelöscht werden.

Die Tonausgabe (GOSUB 400) erfolgt über den TAPE-Ausgang des KC85/3 in einheitlicher Lautstärke. Ist SV = 0, hören Sie keinen Ton. Nutzen Sie einen Monitor mit RGB- oder Videoanschluß, sind Unterscheidungen in den Lautstärken zu hören. Jeder einzelne Ton klingt unabhängig von seiner programmierten Lautstärke aus.

Wollen Sie einen Drucker anschließen, so benötigen Sie ein V24-Modul M003. Die Initialisierung erfolgt beim ersten Start des Bascoders (RUN nach CLOAD"BAC853"). Eine vorhandene V24-Schnittstelle wird auch dann initialisiert, wenn kein Drucker angeschlossen oder dieser "offline" ist. Wurde die Betriebsbereitschaft jedoch einmal erkannt, muß der Drucker bei allen nachfolgenden Druckfunktionen "online" bleiben. Erfolgte RESET, kann im CAOS-Menü so initialisiert werden:

```
%V24I (h v s)
```

Hierin bedeuten: (in Klammern stehen die Standardwerte)

h - die horizontale Vergrößerung für Screencopy (1)
v - vertikale Vergrößerung für Screencopy (1)
s - Anzahl der Space (Leerzeichen) für den linken Rand (10).

Der Drucktreiber arbeitet mit dem Kanal 1 des M003 (V24-Modul; USER-Kanal 2) für alle EPSON-kompatiblen Drucker (K6313/20), aber auch mit der Kleinschreibmaschine S3004/5 u. a.

Für EPSON-kompatible Drucker wird US-ASCII initialisiert. Die Tastenkombination: "shift" und "CLR" oder die Ausgabe von "CHR$(15)" leitet einen Screencopy-Ausdruck des Bildschirmes ein. Dort entsteht, soweit das V24-Modul initialisiert werden konnte, ein negiertes, verschiebbares Fenster. Folgende Steuerfunktionen sind vereinbart: (C- bedeutet: Cursortaste in Richtung ...)

C-oben	=	obere Fenstergrenze nach oben
C-unten	=	untere Fenstergrenze nach unten
C-rechts	=	rechte Fenstergrenze nach rechts
C-links	=	linke Fenstergrenze nach links
shift und C-oben	=	untere Fenstergrenze nach oben
shift und C-unten	=	obere Fenstergrenze nach unten
shift und C-rechts	=	linke Fenstergrenze nach rechts
shift und C-links	=	rechte Fenstergrenze nach links
shift lock	=	Fensterbewegung schnell / langsam
home	=	Fenster invertieren
enter	=	Druckbeginn

Der Grafikdruck kann mit der Taste BRK unterbrochen werden.

Dieser spezielle Druckertreiber wird beim ersten Start nach dem Einladen von BAC853 in den IRM kopiert. Er ist danach nicht mehr Bestandteil des Bascoders. Darum sind Kopien des Bascoders unbedingt vor dem Start mit RUN anzufertigen! (CSAVE"BAC853")

Der Bascoder ist vollständig in Maschinensprache geschrieben. Die Subroutinen der BASIC-Zeilen 10-950 enthalten ausschließlich Sprunganweisungen in das Maschinenprogramm, wo auch alle Parameterübergaben an den Interpreter erfolgen. Diese Sprunganweisungen dürfen vom Nutzer des Bascoders nicht verändert werden und sind deshalb für ihn nicht sichtbar. Die Subroutinen lassen sich im Direkt-Mode nur aufrufen, wenn der rufende Befehl oder eine geeignete Subroutine (GOSUB 100) an erster Stelle der Eingabezeile steht. Dazu ein Beispiel:

```
>GOSUB 25Ø                                        ein Signalton
>FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I                UL ERROR
>GOSUB 1ØØ: FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I     zehn Signaltöne
```

Der Bascoder nutzt eigene Tastaturroutinen und einen veränderten Zufallsgenerator. Bei Verwendung des Hilfsprogramms BASEX ist die Umlenkung der Kassettenein- und -ausgaben auf die KC-Erweiterungsbaugruppe D004 - Floppy Disk garantiert. Das gilt jedoch nicht für BASICODE-Bestände (NF = 4 bis NF = 7).

BASICODE ist auf dem KC85/3 in allen Speicherkonfigurationen lauffähig (16K, 32K, 48K). Für eine Vielzahl von Programmen werden Sie jedoch wenigstens eine Erweiterung um 16K-RAM benötigen. Für Speichererweiterungen oder den Einsatz des HC900 / KC85/2 mit BASIC-Modul M006 gelten die gleichen Bedingungen wie unter Standard-BASIC. Der Bascoder läuft ab CAOS-Version "HC901". Die entsprechenden ROM-Speicherschaltkreise (U2364) tragen die Nummern 601 bzw. 604, für den BASIC-ROM ist Nr. 600 verbindlich.

Probleme beim Lesen von ASCII-Files

Der KC85/3 hat beim Einlesen der ASCII-Files (BASICODE-Format) eine recht mühsame Arbeit zu verrichten und tut sich dabei etwas schwerer als der KC85/4 oder die Computer der Serie KC87. Der Bascoder gleicht dies mit einer automatischen Frequenzerkennung und -einstellung sowie mit einer logischen Korrekturroutine der erkannten Schwingungen aus. Sollten Sie trotzdem Probleme beim Lesen von BASICODE haben, so achten Sie besonders konsequent auf Reinheit und Stellung des Tonkopfes! Der Aufnahmepegel sollte möglichst hoch sein. Zwischen den verschiedenen Recordertypen gibt es ebenfalls Unterschiede.

Die vom Rundfunk der DDR abgestrahlte Testversion des Bascoders zeigte, daß trotz dieser Maßnahmen eine ganze Anzahl von KC85/3-Nutzern selbst die mit dem eigenen Computer und *W auf Kassette geretteten ASCII-Files nicht oder nur mit vielen Fehlern eingelesen bekommen. In den meisten Fällen wird auch statt 2400 Hz nur ca. 1200 Hz erkannt. Die Ursache dafür liegt weder in der Aufzeichnung noch am Recorder oder dem Bascoder. Wir wollen uns an dieser Stelle bei denen

bedanken, die mit großem Zeitaufwand dem Fehler auf der Spur waren. Auch das hat dazu beigetragen, schon in der ersten Ausgabe dieses Buches praktische Hinweise zur Behebung dieser Schwierigkeiten zu geben.

Ein Computerfreund stellte uns im Austausch seinen KC zur Verfügung, denn die Probleme bereitet die Hardware des Kassetteninterfaces. Seine Bandbreite ist für einen Frequenzumfang von 500 Hz bis 2400 Hz ausgelegt. Die Bauelemente haben jedoch immer Toleranzen. Dabei werden vom Computerhersteller auch Abweichungen in der Kennlinie von +-10% eingeplant. So gibt es KC's, die mehr als 2600 Hz verkraften, andere schaffen nicht mal die Mindestanforderung von 2380 Hz. Der Computerhersteller hat das recht geschickt gelöst. Die Kassettenroutinen liegen mit ihren Frequenzen unterhalb der kritischen Werte und nicht wie in der Literatur angegeben bei 600 Hz, 1200 Hz und 2400 Hz. Mit dem Programm "FRQ" haben Sie ein Testprogramm zur Messung der Bandbreite ihres KC's (85/3,4). Es wird mit

```
%LOAD
```

von Kassette eingelesen und anschließend mit

```
%FRQ
```

gestartet. Sie benötigen eine leere Kassette für die Aufzeichnung einer Tonfolge (1). Diese kann dann als Testfile dienen. Mit (2) bekommen Sie Auskunft, ob und mit welcher Sicherheit ihr KC85/3 ASCII-File lesen kann. Eine grafische Darstellung (Echtzeitverarbeitung) erhalten Sie mit (3). Das Programm erklärt sich selbst.

```
Frequenztest

(1) Speichern
(2) Lesen
(3) Grafik
(4) CAOS

==> ?
```

BASICODE verlangt unerbittlich 2400 Hz und es sollte auch eine Reserve vorhanden sein. Sind Sie von diesem Problem betroffen, ist evtl. der Gang zur Vertragswerkstatt nicht zu umgehen. Die notwendigen Arbeiten am KC sind relativ gering und kaum als Fehlerbeseitigung zu bezeichnen (Toleranzen für KC-Verfahren ausreichend).

Zwei Hinweise zur Soforthilfe. Der Fehler tritt nach längerer Gerätelaufzeit weniger stark auf (Temperaturabhängigkeit). Es kann auch die Fähigkeit des Bascoders ausgenutzt werden, sich der erkannten Frequenz anzupassen. So ist es möglich, auf einem Magnetbandgerät mit mehreren Geschwindigkeiten "schnell" aufzunehmen und "langsam" abzuspielen.
Beides eignet sich nicht für den Dauergebrauch. Deshalb an dieser Stelle noch die notwendigen Hardware-Informationen, die der zuständigen Vertragswerkstatt mitgeteilt werden können. Vor eigenmächtigen Eingriffen wird allerdings gewarnt.

Verantwortlich für die Impulsbreitensteuerung des Recordersignals und damit auch für die o. g. Effekte der E/A-Steuerung (IS D3606) ist die Zeitkonstante des Monoflops mit R42/C24. Zur Fehlerbeseitigung wird in der Regel eine Änderung von R42 auf 100k (200k parallel) genügen. Nach unseren Erfahrungen führen weitergehende Arbeiten an der Frequenzbandbegrenzung zwar u. U. zu verbesserten Testwerten, haben aber auf BASICODE keinen Einfluß.

5.2. Bascoder für den KC85/4

Einfache Anleitung

Beim KC85/4 erscheint nach dem Einschalten das bekannte Menü. Der Cursor wird dann auf die Zeile:

```
%BASIC
```

gesetzt und die rechts unten auf der Tastatur liegende Taste (im folgenden ENTER genannt) betätigt. Daraufhin erscheint die Abfrage:

```
HC-BASIC
MEMORY END ? :
```

Bei Bestätigung durch ENTER erfolgt die ergänzende Ausschrift:

```
47854 BYTES FREE
OK
>
```

Bereiten Sie nun das Einlesen des Bascoders von Kassette vor. Alle Bascoder-Versionen des KC85/4 tragen den Namen "BAC854". Sie geben also ein:

```
>CLOAD"BAC854"
```

Diese Eingabe ist zunächst noch nicht mit ENTER zu bestätigen. Die Kassette mit dem Bascoder wird in den Kassettenrecorder eingelegt und durch schnellen Vor- bzw. Rücklauf an die Startstelle (Zählwerk beachten) positioniert. Schalten Sie den Recorder auf Wiedergabe. Beim Ertönen des Starttones wird an der Rechnertastatur die obige Eingabe mit ENTER quittiert. Jetzt kann man das blockweise Einlesen des Bascoders entsprechend den auf dem Bildschirm angezeigten Blocknummern verfolgen. Nach ca. 2 Minuten erscheint:

```
OK
>
```

Der Bascoder ist damit ähnlich wie ein normales BASIC-Programm geladen. Aber noch befindet sich der Rechner nicht im BASICODE-Mode. Hierzu führen Sie den Befehl "RUN" aus. Auf dem Bildschirm entsteht das Startbild mit Hinweisen auf die verwendete Version, KC-Typ, Autoren des Bascoders und dem BASICODE-Menü.

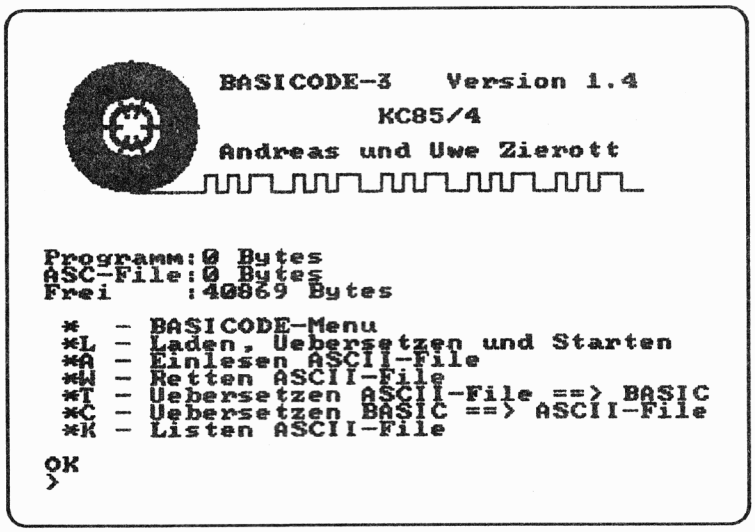


Abbildung 5.14. Startbild KC 85/4

Der Bildschirm zeigt also:

```
Programm:Ø Bytes (Ø Bytes durch BASIC-Programm belegt)
ASC-File:Ø Bytes (Ø Bytes durch ASCII/BASICODE belegt)
Frei:    x Bytes (x Bytes im Arbeitsspeicher frei)
```

Diese Angaben stellen den jeweiligen Status dar. Die Anzahl der freien Bytes klammert den reservierten Stringspeicher aus und ist auch von der verwendeten Bascoder-Version abhängig. Der Bascoder ist jetzt zur Funktionseingabe bereit und bietet neue Kommandos an:

```
  *  - BASICODE - Menu
  *L - Laden, Uebersetzen und Starten
  *A - Einlesen ASCII-File
  *W - Retten ASCII-File
  *T - Uebersetzen ASCII-File in BASIC
  *C - Uebersetzen BASIC in ASCII-File
  *K - Listen ASCII-File
OK
>
```

Ab hier können Sie ein BASICODE-Programm über die Tastatur Ihres Rechners eingeben oder mit dem Kommando

```
*L
```

ein BASICODE-Programm einlesen. Hierzu fordert Sie nach Betätigung der ENTER-Taste die Ausschrift

```
Bitte Band starten
```

auf. Beim Einlesen können Sie den "Programmtext" verfolgen. Nachdem das Programm eingelesen ist, erfolgt automatisch die Übersetzung des BASICODE-Programms in die für den KC85/4 lesbare Form. Im Anschluß daran startet das Programm selbständig. Auf mögliche Fehler bei diesen Vorgängen wird weiter unten eingegangen. Ebenso werden im folgenden die einzelnen Funktionen detaillierter beschrieben.

Achten Sie bei der Kommandoeingabe bitte auch darauf, in welchem Status sich der Rechner befindet. Jede BASICODE-Menüausgabe teilt Ihnen auch mit, ob und in welcher Größe ein

- Programm (BASIC-Programm mit KC-Token)
  und/oder ein
- ASCII-File (BASICODE-Programmtext oder andere ASCII-Datei)

den Speicher Ihres KC's belegt. Die Anzeige des freien Speicherplatzes bezieht sich auf den Programmspeicher und berücksichtigt die Größe der ASCII-Files nicht.

Es kommt zu Fehlerausschriften, wenn Sie Funktionen aufrufen, die sich auf einen nicht vorhandenen Status beziehen. So kann ein ASCII-File nicht mit RUN gestartet werden, und ein KC-Programm läßt sich ohne Übersetzung (*C) nicht mit der Option *W auf Kassette retten.

Hinweise für den Programmierer

Der Bascoder des KC85/4 ist kein neuer BASIC-Dialekt! Den zugelassenen Befehlsvorrat von BASICODE kennen Sie schließlich schon von Ihrem KC-BASIC. Um ein echter BASICODE-Programmierer zu sein, sind jedoch die Regeln strengstens zu beachten. So sind z. B. bei den logischen Operatoren die zugehörigen Ausdrücke einzuklammern, RESTORE existiert nur ohne Zeilennummer und NEXT verlangt die Laufvariable. Ihr Bascoder kann nicht alle Regelverstöße wettmachen, wird Sie aber in der Option *C ggf. daran erinnern.

Viele der nicht zugelassenen BASIC-Befehle wie LINE, LOCATE oder SOUND werden durch die "versteckten" Subroutinen realisiert. Dort finden Sie aber auch Funktionen, die der KC85/4 bislang weder in BASIC noch mit den Routinen seines Betriebssystems zu bieten hatte. Dazu gehören das Aus- bzw. Einschalten der STOP / BRK-Taste, Textausgabe im Grafik-Mode und verschiedene, vom Standard abweichende Funktionen. Betrachten Sie den Bascoder deshalb auch einfach als Erweiterung des Betriebssystems.

Zum Verständnis des Bascoders gehört auch der Umgang mit den Ladebefehlen für die Kassettenarbeit. Ist BASICODE initialisiert, haben Sie auch neue Ladebefehle (*L, *A, *W), um mit dem für alle Bascoder verbindlichen Kassetteninterface arbeiten zu können. Nur auf diese Weise ist der Informationsaustausch zwischen den verschiedensten Computertypen gesichert. Die Ladebefehle des KC-BASIC haben jedoch nicht an Bedeutung verloren.

```
>CLOAD"NAME"
```

können Sie nutzen, um ein beliebiges KC-BASIC-Programm in den Speicher zu laden.

Speziell kann das ein ins KC-BASIC transformiertes BASICODE-Programm sein, das zuvor mit CSAVE"NAME" auf Kassette gesichert wurde. Diese Verfahrensweise ist besonders für die Programmentwicklung anzuraten.

Halten Sie ein BASIC-Programm für geeignet, es den Regeln von BASICODE zu unterwerfen, es also anderen Computertypen zugänglich zu machen, können Sie es zunächst ohne spezielle Programmänderungen dem Bascoder übergeben. Ob der Programmspeicher frei ist, erkennen Sie im Menü. Ist der Wert für "Programm:" > 0 Bytes (Anzeige bei LIST), kann mit

```
>NEW
```

der Programmspeicher gelöscht werden. Ein sofortiger Start nach CLOAD führt mit ziemlicher Sicherheit zu einem Fehler. Dafür gibt es zumindest drei Gründe:

- Der Bascoder benötigt eine Zeile 1000 mit "GOTO 20".
- Der eigentliche Programmstart ist immer in Zeile 1010.
- Die Zeilennummern von 0-950 sind blockiert, sie werden nach RUN vom Bascoder eingeblendet und sind deshalb im Anwenderprogramm wirkungslos.

Hier hilft in jedem Fall das Kommando RENUMBER. Wird es wie üblich im Direkt-Mode verwendet, kann es die Subroutinen des Bascoders nicht beeinflussen.

Als Beispiel sei angenommen, das anzupassende Programm beginne mit Zeile 10 und ende bei Zeile 5830. Um eine geeignete Test-Struktur zu erreichen, tippen Sie eine Zeile 9 ein:

```
>9 A=1ØØ:GOTO2Ø:REM ** Programmname **
```

Die Zeile 9 soll unsere Zeile 1000 werden. Deshalb jetzt:

```
>RENUMBER 9,583Ø,1ØØØ,1Ø
```

Damit haben Sie eine brauchbare Ausgangsbasis auf dem Weg zu einem BASICODE-Programm. Auch später können Sie das Programm oder einzelne Teile von ihm mit RENUMBER neu ordnen und mit einem einfachen Trick sogar die Reihenfolge der Befehlszeilen ändern (geforderte Programmstrukturen beachten). Dazu müssen jedoch zunächst schon einmal alle nicht erlaubten BASIC-Befehle durch die entsprechenden Subroutinen ersetzt bzw. vollständig entfernt worden sein. Nehmen wir an, das vorliegende Programm hat seine DATA-Zeilen nicht zusammenhängend und im Programm verstreut. Durch

```
RENUMBER (ab alte Zeile)(,bis alte Zeile)(,ab neue Zeile),10
```

werden alle Gruppen von DATA-Zeilen nacheinander auf den Bereich ab Zeile 25000 numeriert. Das LIST-Kommando zeigt zwar die erfolgte Veränderung in den Zeilennummern, ihre Plazierung im Programm hat sich jedoch nicht verändert.

Das Programm darf jetzt keinesfalls gestartet oder editiert werden, sonst möglicherweise Systemabsturz !

Um die Zeilen in die gewünschte Reihenfolge zu bringen, sind lediglich zwei BASICODE-Kommandos nacheinander auszuführen:

```
*C - Übersetzen BASIC in ASCII-File
*T - Übersetzen ASCII-File in BASIC
```

Noch einen letzten Hinweis zu RENUMBER. Liegt ein Programm gemäß dem BASICODE-Protokoll vor, sind also bereits Subroutinen der Bascoder eingebunden, sollte auch nicht zum Zwecke der Manipulation unter 1000 numeriert werden.

Es kann viele Gründe geben, BASICODE-Programme als KC-Tokenfile abzuspeichern. Beim KC85/4 mit D004-Erweiterung kommt noch hinzu, daß der Bascoder auch die Diskettenarbeit durch Verwendung der Umlenkroutine BASEX unterstützt.

Befindet sich der KC im BASICODE-Zustand, kann mit dem Befehl:

```
>CSAVE "NAME"
```

das aktuelle Programm-File auf Kassette gespeichert werden. Dabei bleiben die Subroutinen des Bascoders ausgeblendet.

Achtung ! Steht CSAVE nicht am Zeilenanfang, wird das betreffende Programm im Verbund mit dem Bascoder (ohne V24I) gerettet.

Nur für KC-Freaks

Erste Reaktionen auf die Entwicklung der KC-Bascoder zeigen, das nicht wenige Software-Spezialisten bemüht sind, hinter die "Geheimnisse" der Subroutinen zu kommen oder diese gar zu verändern. Tatsächlich ist kein Programm so perfekt, daß es nicht noch wesentlich verbessert werden könnte. Wir sind an Ihren Gedanken und Vorschlägen immer interessiert, weniger dagegen an Rückfragen zu den nicht autorisierten Bascoder-Fassungen. Tatsächlich haben insbesondere die Bascoder der KC's aus Mühlhausen durch die umständliche IRM-Steuerung einen recht komplizierten Aufbau. Von Veränderungen ist deshalb abzuraten, zumal Sie davon ausgehen dürfen, niemals die aktuellste Version zu besitzen.

Ein durchaus "legales Betätigungsfeld" bietet der Bascoder bei Nutzung seiner Subroutinen für maschinennahe, typabhängige Programmierung. Sollen derartige Programme unabhängig von der verwendeten Version sein, sind die Subroutinen wie gewohnt mit GOSUB aufzurufen.

```
>GOSUB 1ØØ : LIST
```

zeigt die versteckten Subroutinen, an denen nicht editiert werden darf.
Die Adressen (CALL*XXXX) unterscheiden sich zwischen den KC-Typen und allen Bascoder-Versionen. Werden sie für besondere Anwendungen trotzdem direkt aufgerufen, könnte mit:

```
>:CSAVE"NAME"          (CSAVE ist nicht am Zeilenanfang)
```

das Programm im Verbund mit dem Bascoder als eine Art Laufzeitmodul auf Kassette gespeichert werden. Solcherart Programme haben mit BASICODE nichts als das "erweiterte" Betriebssystem gemeinsam. Da diese Programme das "Laufzeitmodul" schon enthalten, ist "BAC854" nach dem BASIC-Kaltstart nicht zu laden. Statt dessen ergibt sich folgender Ablauf bis zum Programmstart:

```
>CLOAD"NAME"      (spezielles Programm im Bascoder-Verbund)
>CALL*40D         (Initialisierung Bascoder / Startbild)
>RUN              (Programmstart)
```

Und nun noch ein letzter Hinweis, der nur bedingt etwas mit den eigentlichen Bascoder-Funktionen zu tun hat. Im BASICODE-Menü und in der Beschreibung taucht mehrfach der Begriff ASCII-File statt BASICODE-Programm auf. Ein BASICODE-Programm ist, soweit es mit *C übersetzt oder mit *A geladen wurde, immer ein ASCII-File. Hingegen muß ein ASCII- File nicht zwangsläufig auch ein BASICODE-Programm sein. Denkbar ist dabei nicht nur die Vermittlung von Quelltexten verschiedener Programmiersprachen, sondern auch elektronischer Computermagazine oder anderer Texte.

Befehlsübersicht

Die Auswahl der Funktionen des Bascoders erfolgt durch die Eingabe eines Sternchens "*", dem ein Buchstabe und ENTER folgt. Sie können aber auch die Funktionstasten (F1-F6) nutzen. Über ihre Zweitbelegungen, sie werden hier als F7-FC bezeichnet, erreichen Sie sechs weitere Kommandos.

Als BASICODE-Kommandos (*) sind sowohl Groß- als auch Kleinbuchstaben zugelassen.

Eingabe	Funktion	
*	BASICODE - Menü Es erscheint das Startbild mit allen Funktionen.	
*L (F1)	Laden, Übersetzen und Starten So wird ein BASICODE-Programm von Kassette eingelesen, automatisch in das KC-BASIC übersetzt und gestartet.	
*A (F2)	Einlesen ASCII-File Soll ein Programm oder ein ASCII-File im BASICODE-Format von Kassette in den Speicher geladen werden, wählen Sie diese Funktion. Da nicht übersetzt und gestartet wird, ist sie auch geeignet, Kopien im BASICODE-Format zu erstellen. (*W)	
*W (F3)	Retten ASCII-File Mit dieser Funktion retten Sie ein Programm oder ein anderes ASCII-File im BASICODE-Format auf Kassette.	
*T (F4)	Übersetzen ASCII-File in BASIC Ein im Speicher stehendes Programm im BASICODE-Format wird durch diese Funktion in das KC-BASIC transformiert.	
*C (F5)	Übersetzen BASIC in ASCII-File Ein im Speicher stehendes BASIC-Programm (KC-Format) wird in ein ASCII-File (BASICODE-Format) transformiert.	
*K (F6)	Listen ASCII-File Ein im Speicher stehendes Programm (BASICODE) oder anderes ASCII- File wird auf dem Bildschirm zur Anzeige gebracht.	
(F7)	RUN	- Start eines BASIC-Programms
(F8)	CONT	- Fortsetzung nach BRK
(F9)	LIST	- Listen im KC-Format
(FA)	%REBASIC : CALL*40D	- BASICODE-Restart nach RESET
(FB)	CLOAD"	- Lade BASIC im KC-Format
(FC)	CSAVE"	- Rette BASIC im KC-Format

Jedes "* - Kommando" bewirkt vor seiner Ausführung die Löschung des Bildschirms.

* BASICODE - Menü

Geben Sie nur ein Sternchen ein oder ist das ihm folgende Kommando nicht in der Liste vorhanden, erscheinen das Startbild, die aktuellen Werte der Speicheraufteilung und das oben gezeigte Menü der möglichen Funktionen auf dem Bildschirm.

*L Laden, Übersetzen und Starten (F1)

Die Option *L nutzen Sie, um mit einem Programm, das im BASICODE-Format auf Kassette steht, zu arbeiten.

Zunächst werden Sie aufgefordert, den Recorder zu starten. Erkennt der Bascoder den Vorton, werden von ihm die empfangenen Schwingungen ausgewertet. Die Lese-Routine berücksichtigt die erkannte Frequenz. Kann das Startbyte nicht eindeutig ausgewertet werden, erscheint die Ausschrift:

```
kein BASICODE-Programm !
```

Das Einlesen des ASCII-Files (BASICODE) kann mit geübtem Auge auf der untersten Bildschirmzeile verfolgt werden. Treten offensichtliche Lesefehler auf und Sie wollen das Einlesen abbrechen, so drücken Sie irgendeine Taste.

Ist das Einlesen beendet oder unterbrochen worden, erhalten Sie eine Ausschrift zum Stoppen des Recorders. Ist der verfügbare Speicherplatz zur Ablage des Programms nicht ausreichend, werden Sie mit der Meldung:

```
Speichergrenze erreicht!
```

gewarnt.

Hat der Bascoder alle Zeichen fehlerfrei lesen können, schließen sich automatisch die Transformation in das KC-BASIC (Option *T) und der Programmstart (RUN) an. Anderenfalls erhalten Sie den Hinweis:

```
Lesefehler erkannt !
```

In diesem Fall können Sie eventuell dennoch das ASCII-File mit der *T-Option zu einem editierbaren BASIC-Programm im (KC-Format) machen.

***A Einlesen ASCII-File (F2)**

Diese Option wird verwendet, um ein ASCII-File (im BASICODE-Format) in den Speicher einzulesen. Im Gegensatz zu *L muß das gelesene File nicht zwangsläufig ein echtes BASICODE-Programm sein. Der Anwender übernimmt die Steuerung und kann entscheiden, ob ins KC-Format übersetzt mit *W eine Kopie herzustellen ist oder z. B. mit *K Text zur Anzeige gebracht werden soll. Ansonsten gelten die Erläuterungen zu *L.

***W Retten ASCII-File (F3)**

Zum Kopieren eines ASCII-Files (BASICODE) oder zur Aufzeichnung eines eigenen Programms im BASICODE-Format müssen Sie die *W- Option aufrufen. Zuvor sollte die Kassette positioniert, der Recorder auf "Aufnahme" gestellt und gestartet werden. Ist im Menü das ASCII-File mit 0 Bytes ausgewiesen, erscheint die Warnung:

```
kein ASCII-File im Speicher !
```

Das Auslagern auf Kassette kann in der Statuszeile verfolgt werden. Wollen Sie die Funktion vorzeitig beenden, betätigen Sie irgendeine Taste.

***T Übersetzen ASCII-File in BASIC (F4)**

Das Übersetzen eines BASICODE-Programms (ASCII-File) in das KC-Format führt der Bascoder nach dem fehlerlosen Einladen mit der Option *L automatisch aus. Durch *T kann unabhängig vom Ladevorgang gearbeitet werden. Das ASCII-File bleibt dabei nicht erhalten. Die übersetzten BASIC-Zeilen sind in der Statuszeile sichtbar.

Da der BASIC-Interpreter den Befehl LOG nicht kennt, hat der Bascoder die zusätzliche Aufgabe, bei der Wandlung des ASCII-Files (BASICODE-Programm) in KC-BASIC LOG durch LN zu ersetzen.

***C Übersetzen BASIC in ASCII-File (F5)**

Liegt ein Programm im KC-Format vor, wird es zunächst auf unzulässige Befehle durchsucht. Damit soll verhindert werden, daß BASIC-Anweisungen, die dem KC-Interpreter bekannt, aber nicht im Befehlssatz von BASICODE-3 vereinbart sind, in das ASCII-File gelangen.

Gleichzeitig werden gemäß den Regeln, wie ein ASCII-File auszusehen hat, Leerzeichen (SPACE) gesetzt bzw. entfernt. Logische Operatoren werden auf Klammerung der Ausdrücke getestet. Weitere Kontrollen beziehen sich auf RESTORE (ohne Zeilennummer) und NEXT (mit Laufvariable).

Findet der Bascoder einen Fehler, erhalten Sie die Meldung:

```
unerlaubter BASICODE-Befehl !
```

Oder etwas harmloser:

```
Regelverstoss !
```

Die betreffende BASIC-Zeile wird angezeigt und zur Editierung angeboten. Sie können nun den falschen Befehl verbessern. Nach ENTER befinden Sie sich im Direkt-Mode. Mit *C können Sie einen erneuten Übersetzungsversuch starten.

Nicht jeder Verstoß gegen das BASICODE-Protokoll läßt sich auf diese Weise aufspüren. Wurden Editierungen vorgenommen, macht sich ggf. ein erneuter Programmtest notwendig.

Bei Fehlerfreiheit im ersten Teil beginnt automatisch der zweite Teil - die eigentliche Übersetzung in ein ASCII-File. Dieser Vorgang ist wieder in der Statuszeile zu verfolgen. Dabei wird das BASIC-Programm an das Speicherende geschoben und das ASCII-File Zeile für Zeile an dessen ursprüngliche Stelle gesetzt. Textzeilen, die länger als die zulässigen 60 Zeichen sind, werden als "zu lang" auf dem Bildschirm notiert, der Befehl LN gegen LOG ausgetauscht.

Sind keine Fehlerausschriften zu verzeichnen und arbeitete das Programm im KC-Format fehlerfrei, haben Sie jetzt ein BASICODE-Programm (ASCII), das Sie mit *W auf Kassette schreiben können. Dieses Programm-File kann nun auf jedem Computertyp eingelesen werden, für den es einen Bascoder gibt.

Ist das Programm noch nicht fehlerfrei (Zeilen zu lang), ist es wichtig, Sie transformieren mit *T in das KC-BASIC zurück und führen entsprechende Änderungen durch. Sie gefährden sonst den Programmaustausch zwischen den Computertypen, das wichtigste Anliegen von BASICODE.

*K Listen ASCII-File (F6)

Mit der LIST-Taste oder dem Kommando LIST kann ein Programm im KC-Format, nicht aber ein ASCII-File angezeigt werden. Wollen Sie dieses jedoch sehen (z. B. unmittelbar nach fehlerhaftem Einlesen), so benutzen Sie die Option *K. Fehlerhaft gelesene ASCII-Zeichen, die dadurch einen unzulässigen Charakter erhalten haben (Steuerzeichen), werden als "#" dargestellt. Ab Version 2.0 soll Option *K einen Texteditor aufrufen. Mit ihm werden sowohl Korrekturen im ASCII-File vor der Übersetzung (*T) als auch Erstellung und Editierung BASIC-unabhängiger Files zur Abspeicherung im BASICODE-Format (*W) möglich sein. Es existiert kein gesonderter Befehl zum Löschen eines ASCII-Files. Dies erfolgt automatisch bei jeder Art der Programmeingabe.

Arbeiten mit BASICODE-3-Beständen

Die Bestandsroutinen arbeiten für NF = 0 und NF = 1 mit BASICODE-Beständen gemäß den Festlegungen. Für die Arbeit im KC-Format (NF = 2; NF = 3) werden spezielle Routinen bereitgestellt. Bei Beachtung der zulässigen Zeichen sind u. a. auch Speicherabzüge (z. B. mit Textprogrammen erstellte Daten), insbesondere aber auch KC-Programmlistings und Stringfelder über ein BASICODE-Hilfsprogramm in einen echten BASICODE-Bestand wandelbar. KC-seitig erhalten Sie das geforderte ASCII-Format durch:

```
PRINT#1 Daten
```

Oder für BASIC-Programmlistings:

```
LIST#1"Name"
```

Beachten Sie die Hinweise in Ihrem BASIC-Handbuch. In jedem Fall sollte die Anweisung NULL 0 gegeben werden.

Haben Sie einen BASICODE-Bestand, der mit NF = 3 (KC-Format) auf Kassette gerettet wurde, so kann er ebenfalls außerhalb BASICODE dem KC mit folgenden Befehlen übergeben werden:

```
INPUT#1 Daten
```

Oder für BASIC-Programmlistings:

```
LOAD#1"Name"
```

Auch hier sollte dann die NULL 0 - Anweisung zur Synchronisation gegeben werden. Der Anwender hat zusätzlich zu beachten, daß die mit NF = 3 gespeicherten Daten, genau wie die KC-Kanalausgaben, beim Einlesen im KC-Mode nach dem Endekennzeichen (03H) in den Direkt-Mode des Computers führen. Gegebenenfalls sind eigene Endekennzeichen zu definieren.

Durch wechselseitige Verwendung der BASICODE-Bestandsroutinen stehen dem KC-Nutzer also zusätzliche Verarbeitungsformen offen. Dabei kann der Bascoder entweder Hilfsmittel oder Betriebssystem sein.

Das Laden und Retten von Beständen auf Diskette (NF = 4 bis NF = 7) ist in der vorliegenden Version nicht vorgesehen. Hinweise zum Start bzw. Stop des Recorders werden gegeben. Zwischen den Blöcken braucht der Recorder nicht abgeschaltet zu werden, da das Auffüllen des Textpuffers recht schnell vor sich geht.

Sollen BASICODE-Bestände gelesen werden, prüft das Programm, ob es sich auch wirklich um eine Bestandsdatei handelt. Der Hinweis:

```
kein BASICODE-Bestand!
```

zeigt an, daß der Bascoder keine Bestände vorfindet. Wird ein Bestandsblock fehlerhaft gelesen, folgt die Ausschrift:

```
Lesefehler! zurueck? J/N
```

Kann Ihr Computer den fehlerhaften Block wiederholt nicht lesen, geben Sie "N" ein. Die Daten werden dann trotz Fehler übernommen. Ein gutes BASICODE-Programm wertet diese dann aus.

BASICODE-Restart

Verlassen Sie gewollt BASICODE, etwa um die V24-Schnittstelle zu initialisieren, ist dies mit den gewohnten Kommandos zu erreichen:

```
>BYE          BASICODE in CAOS
%REBASIC      CAOS in BASICODE
```

Der Systemabsturz gehört zu jedem guten KC! Auch BASICODE ändert daran nicht viel. Einige CAOS-Systemfehler werden zwar ausgemerzt, andere mit Sicherheit produziert. Wenn Sie Glück im Unglück haben, verrät Ihnen der Bascoder noch:

```
!Systemabsturz!
```

und Sie haben Ihr Programm nicht verloren. Häufiges Auftreten dieser Erscheinung weist auf einen Gerätefehler oder eine verfälschte Bascoder-Version hin.

War es unumgänglich, den BASICODE-Zustand zu verlassen (RESET), kann dieser u.U. erneut hergestellt werden, ohne den Bascoder nochmals einzuladen. Betätigen Sie dazu die Funktionstaste FA (SHIFT + F4). Bringt das noch keinen Erfolg, ist zunächst:

```
%REBASIC
```

aufzurufen. Erscheint die Bestätigung mit

```
OK
>
```

geben Sie bitte den Befehl

```
>CALL*4ØD
```

ein und lassen Sie ihn durch ENTER ausführen. Somit werden, soweit das noch möglich ist, die Systemzeiger auf BASICODE gesetzt. Sie erkennen den ordnungsgemäßen RESTART am Startbild.

Besonderheiten des KC85/4

Die Bedienoberfläche des Bascoders wurde für alle KC-Typen einheitlich gestaltet. Die Gleichheit der Kommandos und Ähnlichkeiten in ihrer Eingabe zwischen KC und ZX-Spectrum sind ebenfalls gewollt. Wir hoffen, daß diese Tatsache zu einer schnellen Verbreitung und Nutzung von BASICODE beiträgt.

Der Textschirm des KC85/4 besteht aus 32 Zeilen mit je 40 Zeichen. Die Vollgrafik (256*320) nutzt der Bascoder mit 240*320 Pixel aus. Im Gegensatz zum KC85/3 finden die System-Unterprogramme in den Grafik-Subroutinen Verwendung.

Der ASCII-Zeichensatz des KC85/4 weicht vom Standard (US-ASCII) ab. Die Codes 91-93 und 96 liegen in der ersten Tastaturebene, geben aber andere Zeichen auf dem Bildschirm aus. Auch die Codes 123-126 nutzt der KC zur Darstellung anderer Zeichen (ä,ö,ü,ß). Sie sind aber nur über die zweite Tastaturebene zu erreichen.

Ist innerhalb von BASICODE-Programmen ein Zeichen zu löschen, so ist dazu die Taste CLR vorgesehen. Sie gibt den Code 127 ab. Im EDIT-Modus hat sie ihre alte Bedeutung verloren.

Die Forderungen der Subroutine 650 wurden realisiert. Schrift kann auch unabhängig von den Textzeilen grafisch geschrieben und wieder gelöscht werden.

Die Tonausgabe (GOSUB 400) erfolgt über den TAPE-Ausgang des KC85/4 in einheitlicher Lautstärke. Ist SV = 0, hören Sie keinen Ton. Nutzen Sie einen Monitor mit RGB- oder Videoanschluß, sind Unterscheidungen in den Lautstärken zu hören. Jeder einzelne Ton klingt unabhängig von seiner programmierten Lautstärke aus.

Wollen Sie einen Drucker anschließen, so benötigen Sie ein V24-Modul M003. Die Initialisierung erfolgt beim ersten Start des Bascoders (RUN nach CLOAD"BAC854"). Eine vorhandene V24-Schnittstelle wird auch dann initialisiert, wenn kein Drucker angeschlossen oder dieser "offline" ist. Wurde die Betriebsbereitschaft jedoch einmal erkannt, muß der Drucker bei allen nachfolgenden Druckfunktionen "online" bleiben. Erfolgte RESET, tritt an die Stelle des "BASICODE-Drucktreibers" wieder der KC85/4-Standardtreiber. Wollen Sie weiter mit dem Grafikfenster arbeiten oder die Parameter verstellen, so kann im CAOS-Menü so initialisiert werden:

```
%V24I (h v s)
```

Hierin bedeuten: (in Klammern stehen die Standardwerte)

h - die horizontale Vergrößerung für Screencopy (1)
v - vertikale Vergrößerung für Screencopy (1)
s - Anzahl der Space (Leerzeichen) für den linken Rand (10).

Der Drucktreiber arbeitet mit dem Kanal 1 des M003 (V24-Modul; USER-Kanal 2) für alle EPSON-kompatiblen Drucker (K6313/20), aber auch mit der Kleinschreibmaschine S3004/5 u. a.

Für EPSON-kompatible Drucker wird US-ASCII initialisiert. Die Tastenkombination: "shift" und "CLR" oder die Ausgabe von "CHR$(15)" leitet einen Screencopy-Ausdruck des Bildschirmes ein. Dort entsteht, soweit das V24-Modul initialisiert werden konnte, ein negiertes, verschiebbares Fenster. Folgende Steuerfunktionen sind vereinbart: (C- bedeutet: Cursortaste in Richtung ...)

C-oben	=	obere Fenstergrenze nach oben
C-unten	=	untere Fenstergrenze nach unten
C-rechts	=	rechte Fenstergrenze nach rechts
C-links	=	linke Fenstergrenze nach links
shift und C-oben	=	untere Fenstergrenze nach oben
shift und C-unten	=	obere Fenstergrenze nach unten
shift und C-rechts	=	linke Fenstergrenze nach rechts
shift und C-links	=	rechte Fenstergrenze nach links
shift lock	=	Fensterbewegung schnell / langsam
home	=	Fenster invertieren
enter	=	Druckbeginn

Der Grafikdruck kann mit der Taste BRK unterbrochen werden.

Dieser spezielle Druckertreiber wird beim ersten Start nach dem Einladen von BAC854 in den IRM kopiert. Er ist danach nicht mehr Bestandteil des Bascoders. Darum sind Kopien des Bascoders unbedingt vor dem Start mit RUN anzufertigen! (CSAVE"BAC854")

Der Bascoder ist vollständig in Maschinensprache geschrieben. Die Subroutinen der BASIC-Zeilen 10-950 enthalten ausschließlich Sprunganweisungen in das Maschinenprogramm, wo auch alle Parameterübergaben an den Interpreter erfolgen. Diese Sprunganweisungen dürfen vom Nutzer des Bascoders nicht verändert werden und sind deshalb für ihn nicht sichtbar. Die Subroutinen lassen sich im Direkt-Mode nur aufrufen, wenn der rufende Befehl oder eine geeignete Subroutine (GOSUB 100) an erster Stelle der Eingabezeile steht. Dazu ein Beispiel:

```
>GOSUB 25Ø                                    ein Signalton
>FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I            UL ERROR
>GOSUB 1ØØ: FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I  zehn Signaltöne
```

Der Bascoder nutzt eigene Tastaturroutinen und einen veränderten Zufallsgenerator. Bei Verwendung des Hilfsprogramms BASEX ist die Umlenkung der Kassettenein- und -ausgaben auf die KC-Erweiterungsbaugruppe D004-Floppy Disk garantiert. Das gilt jedoch nicht für BASICODE-Bestände (NF = 4 bis NF = 7).

5.3. Bascoder für den KC87

Einfache Anleitung

Beim KC87 erscheint nach dem Einschalten der bekannte Systemstart:

```
robotron Z 9ØØ1
OS
>
```

Sie geben jetzt ein:

```
>BASIC
```

und betätigen die rechts unten auf der Tastatur liegende ENTER-Taste.
Daraufhin erscheint die Abfrage:

```
HC-BASIC
MEMORY END ? :
```

Bei Bestätigung durch ENTER erfolgt - sofern keine Speichererweiterung vorliegt - die ergänzende Ausschrift:

```
15Ø86 BYTES FREE
OK
>
```

Bereiten Sie nun das Einlesen des Bascoders von Kassette vor. Alle Bascoder-Versionen des KC 87 tragen den Namen "BAC87". Sie geben also ein:

```
>CLOAD"BAC87"
```

Diese Eingabe ist zunächst noch nicht mit ENTER zu bestätigen. Die Kassette mit dem Bascoder wird in den Kassettenrecorder eingelegt und durch schnellen Vor- bzw. Rücklauf an die Startstelle (Zählwerk beachten) positioniert. Schalten Sie den Recorder auf Wiedergabe. Beim Ertönen des Starttones wird an der Rechnertastatur die obige Eingabe mit ENTER quittiert. Bei eingeschaltetem Cursor kann das blockweise Einlesen des Bascoders auf dem Bildschirm verfolgt werden. Nach ca. 2 Minuten erscheint:

```
OK
>
```

Der Bascoder ist damit ähnlich wie ein normales BASIC-Programm geladen. Aber noch befindet sich der Rechner nicht im BASICODE-Mode. Hierzu führen Sie den Befehl

```
>RUN
```

aus. Auf dem Bildschirm entsteht das Startbild mit Hinweisen auf die verwendete Version, KC-Typ, Autoren des Bascoders und dem BASICODE-Menü.

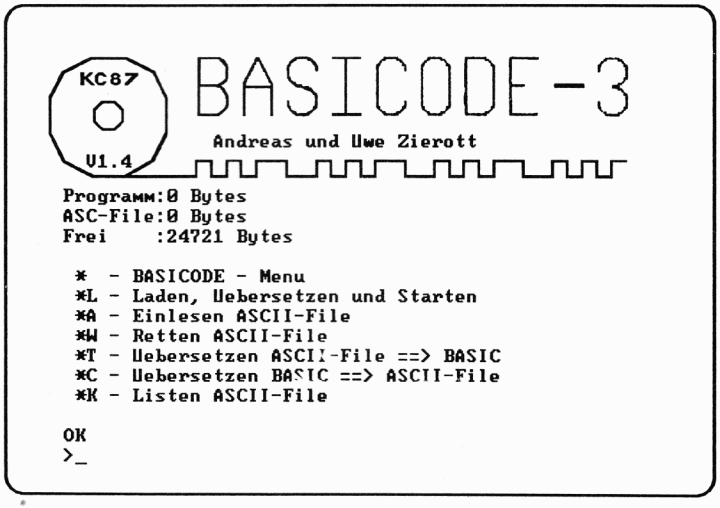


Abbildung 5.15. Startbild KC 87

Der Bildschirm zeigt also:

```
Programm:Ø Bytes (Ø Bytes durch BASIC-Programm belegt)
ASC-File:Ø Bytes (Ø Bytes durch ASCII/BASICODE belegt)
Frei     :x Bytes (x Bytes im Arbeitsspeicher frei)
```

Diese Angaben stellen den jeweiligen Status dar. Die Anzahl der freien Bytes klammert den reservierten Stringspeicher aus und ist auch von der verwendeten Bascoder-Version und den Speichererweiterungen abhängig. Der Bascoder ist jetzt zur Funktionseingabe bereit und bietet neue Kommandos an:

```
*  - BASICODE - Menu
*L - Laden, Uebersetzen und Starten
*A - Einlesen ASCII-File
*W - Retten ASCII- File
*T - Uebersetzen ASCII-File in BASIC
*C - Uebersetzen BASIC in ASCII-File
*K - Listen ASCII-File
OK
>
```

Ab hier können Sie ein BASICODE-Programm über die Tastatur Ihres Rechners eingeben oder mit dem Kommando:

```
*L
```

ein BASICODE-Programm einlesen. Hierzu fordert Sie nach Betätigung der ENTER-Taste die Ausschrift:

```
Bitte Band starten
```

auf. Beim Einlesen können Sie den "Programmtext" verfolgen. Nachdem das Programm eingelesen ist, erfolgt automatisch die Übersetzung des BASICODE-Programms in die für den KC87 lesbare Form. Im Anschluß daran startet das BASIC-Programm selbständig. Auf mögliche Fehler bei diesen Vorgängen wird weiter unten eingegangen. Ebenso werden im folgenden die einzelnen Funktionen detaillierter beschrieben.

Achten Sie bei der Kommandoeingabe bitte auch darauf, in welchem Status sich der Rechner befindet. Jede BASICODE-Menüausgabe teilt Ihnen auch mit, ob und in welcher Größe ein

- Programm (BASIC-Programm mit KC-Token) und/oder ein
- ASCII-File (BASICODE-Programmtext oder andere ASCII-Datei)

den Speicher Ihres KC's belegt. Die Anzeige des freien Speicherplatzes bezieht sich auf den Programmspeicher und berücksichtigt die Größe der ASCII-Files nicht.

Es kommt zu Fehlerausschriften, wenn Sie Funktionen aufrufen, die sich auf einen nicht vorhandenen Bascoder-Status beziehen. So kann ein ASCII-File nicht mit RUN gestartet werden, und ein KC-Programm läßt sich ohne Übersetzung (*C) nicht mit der Option *W auf Kassette retten.

Hinweise für den Programmierer

Der Bascoder des KC87 ist kein neuer BASIC-Dialekt! Den zugelassenen Befehlsvorrat von BASICODE kennen Sie schließlich schon von Ihrem KC-BASIC. Um ein echter BASICODE-Programmierer zu sein, sind jedoch die Regeln strengstens zu beachten. So sind z. B. bei den logischen Operatoren die zugehörigen Ausdrücke einzuklammern, RESTORE existiert nur ohne Zeilennummer und NEXT verlangt die Laufvariable. Ihr Bascoder kann nicht alle Regelverstöße wettmachen, wird Sie aber in der Option *C ggf. daran erinnern.

Viele der nicht zugelassenen BASIC-Befehle wie BEEP, PRINT AT oder CLS werden durch die "versteckten" Subroutinen realisiert. Dort finden Sie aber auch Funktionen, die der KC 87 bislang weder in BASIC noch mit den Routinen seines Betriebssystems zu bieten hatte. Dazu gehören das Aus- bzw. Einschalten der STOP / BRK-Taste und die Erweiterungen um Sound und "Grafik".

Betrachten Sie den Bascoder deshalb auch einfach als Erweiterung des Betriebssystems.

Zum Verständnis des Bascoders gehört auch der Umgang mit den Ladebefehlen für die Kassettenarbeit. Ist BASICODE initialisiert, haben Sie auch neue Ladebefehle (*L, *A, *W), um mit dem für alle Bascoder verbindlichen Kassetteninterface arbeiten zu können. Nur auf diese Weise ist der Informationsaustausch zwischen den verschiedensten Computertypen gesichert. Die Ladebefehle des KC-BASIC haben jedoch nicht an Bedeutung verloren.

```
>CLOAD"NAME"
```

können Sie nutzen, um ein beliebiges KC-BASIC-Programm in den Speicher zu laden.

Speziell kann das ein ins KC-BASIC transformiertes BASICODE-Programm sein, das zuvor mit CSAVE"NAME" auf Kassette gesichert wurde. Diese Verfahrensweise ist besonders für die Programmentwicklung anzuraten.

Halten Sie ein BASIC-Programm für geeignet, es den Regeln von BASICODE zu unterwerfen, es also anderen Computertypen zugänglich zu machen, können Sie es zunächst ohne spezielle Programmänderungen dem Bascoder übergeben. Ob der Programmspeicher frei ist, erkennen Sie im Menü. Ist der Wert für "Programm:" > 0 Bytes (Anzeige bei LIST), kann mit

```
>NEW
```

der Programmspeicher gelöscht werden. Ein sofortiger Start nach CLOAD führt mit ziemlicher Sicherheit zu einem Fehler. Dafür gibt es zumindest drei Gründe:

- Der Bascoder benötigt eine Zeile 1000 mit "GOTO 20".
- Der eigentliche Programmstart ist immer in Zeile 1010.
- Die Zeilennummern von 0-950 sind blockiert, sie werden nach RUN vom Bascoder eingeblendet und sind deshalb im Anwenderprogramm wirkungslos.

Hier hilft in jedem Fall das Kommando RENUM. Wird es wie üblich im Direkt-Mode verwendet, kann es die Subroutinen des Bascoders nicht beeinflussen.

Als Beispiel sei angenommen, das anzupassende Programm beginne mit Zeile 10 und ende bei Zeile 5830. Um eine geeignete Teststruktur zu erreichen, tippen Sie eine Zeile 9 ein.

```
>9 A=1ØØ:GOTO2Ø:REM ** Programmname **
```

Die Zeile 9 soll unsere Zeile 1000 werden. Deshalb jetzt:

```
>RENUM 1ØØØ
```

Damit haben Sie eine brauchbare Ausgangsbasis auf dem Weg zu einem BASICODE-Programm. Auch später können Sie das komplette Programm oder ab einer bestimmten Zeilennummer neu ordnen. Bei Verwendung eines KC87 (produziert bis Anfang 1988) eines BASIC-Moduls mit alter Erweiterungsliste (außerhalb BASICODE) oder eines KC85/3,4 (Arbeitsgemeinschaft) steht Ihnen der Befehl RENUMBER zur Verfügung. Nur so kann auch durch einen einfachen Trick die Reihenfolge der Befehlszeilen verändert werden (geforderte Programmstrukturen beachten). Dazu müssen jedoch zunächst schon einmal alle nicht erlaubten BASIC-Befehle durch die entsprechenden Subroutinen ersetzt bzw. vollständig entfernt worden sein. Nehmen wir an, das vorliegende Programm hat seine DATA-Zeilen nicht zusammenhängend und im Programm verstreut. Durch:

```
RENUMBER (ab alte Zeile) (,bis alte Zeile) (,ab neue Zeile) ,10
```

werden alle Gruppen von DATA-Zeilen nacheinander auf den Bereich ab Zeile 25000 numeriert. Das LIST-Kommando zeigt zwar die erfolgte Veränderung in den Zeilennummern, ihre Plazierung im Programm hat sich jedoch nicht verändert.

Das Programm darf jetzt keinesfalls gestartet oder editiert werden, sonst möglicherweise Systemabsturz !

Um die Zeilen in die gewünschte Reihenfolge zu bringen, sind lediglich zwei BASICODE-Kommandos nacheinander auszuführen:

```
*C  -  Übersetzen BASIC in ASCII-File
*T  -  Übersetzen ASCII-File in BASIC
```

Noch einen letzten Hinweis zu RENUMBER. Liegt ein Programm gemäß dem BASICODE-Protokoll vor, sind also bereits Subroutinen der Bascoders eingebunden, sollte auch nicht zum Zwecke der Manipulation unter 1000 numeriert werden.

Es kann viele Gründe geben, BASICODE-Programme als KC-Tokenfile abzuspeichern. Befindet sich der KC im BASICODE-Zustand, kann mit dem Befehl:

```
>CSAVE "NAME"
```

das aktuelle Programm-File auf Kassette gespeichert werden. Dabei bleiben die Subroutinen des Bascoders ausgeblendet.

Achtung ! Steht CSAVE nicht am Zeilenanfang, wird das betreffende Programm im Verbund mit dem Bascoder gerettet.

Nur für KC-Freaks

Erste Reaktionen auf die Entwicklung der KC-Bascoder zeigen, das nicht wenige Software-Spezialisten bemüht sind, hinter die "Geheimnisse" der Subroutinen zu kommen oder diese gar zu verändern. Tatsächlich ist kein Programm so perfekt, daß es nicht noch wesentlich verbessert werden könnte. Wir sind an Ihren Gedanken und Vorschlägen immer interessiert, weniger dagegen an Rückfragen zu den nicht autorisierten Bascoder-Fassungen.

Auch von Veränderungen zwecks Anpassung an die ersten Modul-Versionen des BASIC-Interpreters ist abzuraten. Sinnvoller wäre es da schon, in einer Vertragswerkstatt die EPROMs neu brennen zu lassen, zumal Sie davon ausgehen dürfen, niemals die aktuellste Version zu besitzen.

Ein durchaus "legales Betätigungsfeld" bietet der Bascoder bei Nutzung seiner Subroutinen für maschinennahe, typabhängige Programmierung. Sollen derartige Programme unabhängig von der verwendeten Version sein, sind die Subroutinen wie gewohnt mit GOSUB aufzurufen.

```
>GOSUB 1ØØ : LIST
```

zeigt die versteckten Subroutinen, an denen nicht editiert werden darf.
Die Adressen (CALL*XXXX) unterscheiden sich zwischen den KC-Typen und allen Bascoder-Versionen. Die Programme sind mit

```
>CSAVE "NAME"
```

zu retten und mit

```
>CLOAD "NAME"
```

zu laden. Zuvor mußte der Bascoder gestartet werden. Wie auch bei der KC85-Serie können Bascoder und Programm mit einem Befehl gesichert werden.

```
>:CSAVE "NAME"
```

Die Möglichkeiten nach dem Laden sind jedoch eingeschränkt, weshalb diese Methode für den KC87 kaum in Frage kommt.

```
>CLOAD "NAME"    (Bascoder und Programm im Verbund)
>RUN             (Initialisierung Bascoder/Startbild)
>*C              (BASIC= = >ASCII-File)
>RUN             (Programmstart)
```

Es ist zu ersehen, daß hierbei keinerlei Vorteil entsteht. Das Programm darf auch keine maschinennahen Befehle enthalten. Sie würden bei *C abgewiesen werden. Und nun noch ein letzter Hinweis, der nur bedingt etwas mit den eigentlichen Bascoder-Funktionen zu tun hat. Im BASICODE-Menü und in der Beschreibung taucht mehrfach der Begriff ASCII-File statt BASICODE-Programm auf. Ein BASICODE-Programm ist, soweit es mit *C übersetzt oder mit *A geladen wurde, immer ein ASCII-File. Hingegen muß ein ASCII-File nicht zwangsläufig auch ein BASICODE-Programm sein. Denkbar ist dabei nicht nur die Vermittlung von Quelltexten verschiedener Programmiersprachen, sondern auch "elektronische Computermagazine" oder andere Texte.

Befehlsübersicht

Die Auswahl der Funktionen des Bascoders erfolgt durch die Eingabe eines Sternchens "*", dem ein Buchstabe und ENTER folgt. Als BASICODE-Kommandos (*) sind sowohl Groß- als auch Kleinbuchstaben zugelassen.

Eingabe	Funktion
*	BASICODE - Menü Es erscheint das Startbild mit allen Funktionen.
*L	Laden, Übersetzen und Starten So wird ein BASICODE-Programm von Kassette eingelesen, automatisch in das KC-BASIC übersetzt und gestartet.
*A	Einlesen ASCII-File Soll ein Programm oder ein anderes ASCII-File im BASICODE-Format von Kassette in den Speicher geladen werden, wählen Sie diese Funktion. Da nicht übersetzt und gestartet wird, ist sie auch geeignet, Kopien im BASICODE-Format zu erstellen (*W).
*W	Retten ASCII-File Mit dieser Funktion retten Sie ein Programm oder ein anderes ASCII-FILE im BASICODE-Format auf Kassette.
*T	Übersetzen ASCII-File in BASIC Ein im Speicher stehendes Programm im BASICODE-Format wird durch diese Funktion in das KC-BASIC transformiert.
*C	Übersetzen BASIC in ASCII-File Ein im Speicher stehendes BASIC-Programm (KC-Format) wird in ein ASCII-File (BASICODE-Format) transformiert.
*K	Listen ASCII-File Ein im Speicher stehendes Programm (BASICODE) oder anderes ASCII-File wird auf dem Bildschirm zur Anzeige gebracht.

Jedes "* - Kommando" bewirkt vor seiner Ausführung die Löschung des Bildschirmes.

* BASICODE - Menü

Geben Sie nur ein Sternchen ein oder ist das ihm folgende Kommando nicht in der Liste vorhanden, erscheinen das Startbild, die aktuellen Werte der Speicheraufteilung und das oben gezeigte Menü der möglichen Funktionen auf dem Bildschirm.

*L Laden, Übersetzen und Starten

Die Option *L nutzen Sie, um mit einem Programm, das im BASICODE-Format auf Kassette steht, zu arbeiten.

Zunächst werden Sie aufgefordert, den Recorder zu starten. Erkennt der Bascoder den Vorton, werden von ihm die empfangenen Schwingungen ausgewertet. Die Lese-Routine berücksichtigt die erkannte Frequenz. Kann das Startbyte nicht eindeutig ausgewertet werden, erscheint die Ausschrift:

```
kein BASICODE-Programm !
```

Das Einlesen des ASCII-Files (BASICODE) kann mit geübtem Auge auf der untersten Bildschirmzeile verfolgt werden. Treten offensichtliche Lesefehler auf und Sie wollen das Einlesen abbrechen, so drücken Sie irgendeine Taste.

Ist das Einlesen beendet oder unterbrochen worden, erhalten Sie eine Ausschrift zum Stoppen des Recorders. Ist der verfügbare Speicherplatz zur Ablage des Programms nicht ausreichend, werden Sie mit der Meldung:

```
Speichergrenze erreicht!
```

gewarnt.

Hat der Bascoder alle Zeichen fehlerfrei lesen können, schließen sich automatisch die Transformation in das KC-BASIC (Option *T) und der Programmstart (RUN) an. Anderenfalls erhalten Sie den Hinweis:

```
Lesefehler erkannt !
```

In diesem Fall können Sie eventuell dennoch das ASCII-File mit der *T-Option zu einem editierbaren BASIC-Programm im (KC-Format) machen.

*A Einlesen ASCII-File

Diese Option wird verwendet, um ein ASCII-File (im BASICODE-Format) in den Speicher einzulesen. Im Gegensatz zu *L muß das gelesene File nicht zwangsläufig ein echtes BASICODE-Programm sein. Der Anwender übernimmt die Steuerung und kann entscheiden, ob ins KC-Format übersetzt mit *W eine Kopie herzustellen ist oder z. B. mit *K Text zur Anzeige gebracht werden soll. Ansonsten gelten die Erläuterungen zu *L.

*W Retten ASCII-File

Zum Kopieren eines ASCII-Files (BASICODE) oder zur Aufzeichnung eines eigenen Programms im BASICODE-Format, müssen Sie die *W-Option aufrufen. Zuvor sollte die Kassette positioniert, der Recorder auf "Aufnahme" gestellt und gestartet werden. Ist im Menü das ASCII-File mit 0 Bytes ausgewiesen, erscheint die Warnung:

```
kein ASCII-File im Speicher !
```

Das Auslagern auf Kassette kann in der Statuszeile verfolgt werden. Wollen Sie die Funktion vorzeitig beenden, betätigen Sie irgendeine Taste.

*T Übersetzen ASCII-File in BASIC

Das Übersetzen eines BASICODE-Programms (ASCII-File) in das KC-Format führt der Bascoder nach dem fehlerlosen Einladen mit der Option *L automatisch aus. Durch *T kann unabhängig vom Ladevorgang gearbeitet werden. Das ASCII-File bleibt dabei nicht erhalten. Die übersetzten BASIC-Zeilen werden auf dem Bildschirm gelistet.

Da der BASIC-Interpreter den Befehl LOG nicht kennt, hat der Bascoder die zusätzliche Aufgabe, bei der Wandlung des ASCII-Files (BASICODE-Programm) in KC-BASIC LOG durch LN zu ersetzen.

*C Übersetzen BASIC in ASCII-File

Liegt ein Programm im KC-Format vor, wird es zunächst auf unzulässige Befehle durchsucht. Damit soll verhindert werden, daß BASIC-Anweisungen, die dem KC-Interpreter bekannt, aber nicht im Befehlssatz von BASICODE-3 vereinbart sind, in das ASCII-File gelangen.

Gleichzeitig werden gemäß der Regeln, wie ein ASCII-File auszusehen hat, Leerzeichen (SPACE) gesetzt bzw. entfernt. Logische Operatoren werden auf Klammerung der Ausdrücke getestet. Weitere Kontrollen beziehen sich auf RESTORE (ohne Zeilennummer) und NEXT (mit Laufvariable).

Findet der Bascoder einen Fehler, erhalten Sie die Meldung:

```
unerlaubter BASICODE-Befehl !
```

Oder etwas harmloser:

```
Regelverstoss !
```

Die betreffende BASIC-Zeile wird angezeigt und zur Editierung angeboten. Sie können nun den falschen Befehl verbessern. Nach ENTER befinden Sie sich im Direkt-Mode. Mit *C können Sie einen erneuten Übersetzungsversuch starten.

Nicht jeder Verstoß gegen das BASICODE-Protokoll läßt sich auf diese Weise aufspüren. Wurden Editierungen vorgenommen, macht sich ggf. ein erneuter Programmtest notwendig.

Bei Fehlerfreiheit im ersten Teil beginnt automatisch der zweite Teil - die eigentliche Übersetzung in ein ASCII-File. Dieser Vorgang ist wieder in der Statuszeile zu verfolgen. Dabei wird das BASIC-Programm an das Speicherende geschoben und das ASCII-File Zeile für Zeile an dessen ursprüngliche Stelle gesetzt. Textzeilen, die länger als die zulässigen 60 Zeichen sind, werden als "zu lang" auf dem Bildschirm notiert, der Befehl LN gegen LOG ausgetauscht.

Sind keine Fehlerausschriften zu verzeichnen und arbeitete das Programm im KC-Format fehlerfrei, haben Sie jetzt ein BASICODE-Programm (ASCII), das Sie mit *W auf Kassette schreiben können. Dieses Programm-File kann nun auf jedem Computertyp eingelesen werden, für den es einen Bascoder gibt.

Ist das Programm noch nicht fehlerfrei (Zeilen zu lang), ist es wichtig, Sie transformieren mit *T in das KC-BASIC zurück und führen entsprechende Änderungen durch. Sie gefährden sonst den Programmaustausch zwischen den Computertypen, das wichtigste Anliegen von BASICODE.

*K Listen ASCII-File

Mit der LIST-Taste oder dem Kommando LIST kann ein Programm im KC-Format, nicht aber ein ASCII-File angezeigt werden. Wollen Sie dieses jedoch sehen (z. B. unmittelbar nach fehlerhaftem Einlesen), so benutzen Sie die Option *K. Fehlerhaft gelesene ASCII-Zeichen, die dadurch einen unzulässigen Charakter erhalten haben (Steuerzeichen), werden als "#" dargestellt. Ab Version 2.0 soll Option *K einen Texteditor aufrufen. Mit ihm werden sowohl Korrekturen im ASCII-File vor der Übersetzung (*T) als auch Erstellung und Editierung BASIC-unabhängiger Files zur Abspeicherung im BASICODE-Format (*W) möglich sein. Es existiert kein gesonderter Befehl zum Löschen eines ASCII-Files. Dies erfolgt automatisch bei jeder Art der Programmeingabe.

Arbeiten mit BASICODE-3-Beständen

Die Bestandsroutinen (für File und Dateien) arbeiten für NF = 0 und NF = 1 mit BASICODE-Beständen gemäß dem Protokoll. Für die Arbeit im KC-Format (NF = 2; NF = 3) werden spezielle Routinen bereitgestellt. Somit sind z. B. auch KC87-Programmlistings über ein BASICODE-Hilfsprogramm in einen echten BASICODE-Bestand wandelbar. Mit KC- Befehlen erhalten Sie das geforderte ASCII-Format durch:

```
LIST#1"Name"
```

Beachten Sie die Hinweise in Ihrem BASIC-Handbuch. In jedem Fall sollte die Anweisung NULL 0 gegeben werden.

Haben Sie einen BASICODE-Bestand, der mit NF = 3 (KC-Format) auf Kassette gerettet wurde, so kann er ebenfalls außerhalb BASICODE dem KC mit folgendem Befehl übergeben werden:

```
LOAD#1"Name"
```

Auch hier sollte dann die NULL 0 - Anweisung zur Synchronisation gegeben werden. Der Anwender hat zusätzlich zu beachten, daß die mit NF = 3 gespeicherten Daten, genau wie die KC-Kanalausgaben, beim Einlesen im KC-Mode nach dem Endekennzeichen (03H) in den Direkt-Mode des Computers führen. Gegebenenfalls sind für den KC-Mode Hilfsprogramme zu schreiben, um die neuen Möglichkeiten der BASICODE-Bestandsarbeit auch auf dieser Ebene ausnutzen zu können.

Das Laden und Retten von Beständen auf Diskette (NF = 4 bis NF = 7) ist in der vorliegenden Version nicht vorgesehen. Hinweise zum Start bzw. Stop des Recorders werden gegeben. Zwischen den Blöcken braucht der Recorder nicht abgeschaltet zu werden, da das Auffüllen des Textpuffers recht schnell vor sich geht.

Sollen BASICODE-Bestände gelesen werden, prüft das Programm, ob es sich auch wirklich um eine Bestandsdatei handelt. Der Hinweis:

```
kein BASICODE-Bestand!
```

zeigt an, daß der Bascoder keine Bestände vorfindet. Wird ein Bestandsblock fehlerhaft gelesen, folgt die Ausschrift:

```
Lesefehler! zurueck? J/N
```

Kann Ihr Computer den fehlerhaften Block wiederholt nicht lesen, geben Sie "N" ein. Die Daten werden dann trotz Fehler übernommen. Ein gutes BASICODE-Programm wertet diese dann aus.

BASICODE-Restart

Der Systemabsturz gehört zu jedem guten KC! Auch BASICODE ändert daran nicht viel. Wenn Sie Glück im Unglück haben, verrät Ihnen der Bascoder noch:

```
!Systemabsturz!
```

und Sie haben Ihr Programm nicht verloren. Häufiges Auftreten dieser Erscheinung weist auf einen Gerätefehler oder eine verfälschte Bascoder-Version hin.

War es unumgänglich, den BASICODE-Zustand zu verlassen (RESET), kann dieser u. U. erneut hergestellt werden, ohne den Bascoder nochmals einzuladen. Dazu ist zunächst

```
>WBASIC
```

aufzurufen. Erscheint dann die Bestätigung mit

```
OK
>
```

geben Sie bitte den Befehl

```
>CALL*4ØD
```

ein und lassen Sie ihn durch ENTER ausführen. Somit werden, soweit das noch möglich ist, die Systemzeiger auf BASICODE gesetzt. Sie erkennen den ordnungsgemäßen RESTART am Startbild.

Besonderheiten des KC87

Die Bedienoberfläche des Bascoders wurde für alle KC-Typen einheitlich gestaltet. Die Gleichheit der Kommandos und Ähnlichkeiten ihrer Eingabe zwischen KC und ZX-Spectrum sind ebenfalls gewollt. Wir hoffen, daß diese Tatsache zu einer schnellen Verbreitung und Nutzung von BASICODE beiträgt.

Der Textschirm des KC87 besteht aus 24 Zeilen mit je 40 Zeichen. Auch in seinem Grafik-Modus (Pseudografik) erreicht er keine höhere Auflösung. BASICODE wertet diesbezüglich Ihren Computer auf. Sie verfügen nunmehr über einen grafischen Bildschirm der doppelten Auflösung, wovon 48*64 "Pixel" genutzt werden können. Somit sind Ihnen auch Programme erschlossen, die sonst nur auf grafikfähigen Computern liefen. Sicher, Unterschiede im Detail sind nicht zu übersehen. Dafür sind Ihre Bilder aber schneller fertig!

Die ASCII-Codes 91-93 und 123-126 geben die Zeichen entsprechend der BASICODE-Vorschriften aus (US-ASCII). Sie sind jedoch von der Tastatur aus nicht zu erreichen.

Inverse Schrift kann auf dem KC87 nicht erzeugt werden. Die Subroutine zur auffälligen Anzeige (GOSUB 150) setzt deshalb ein Karo vor und hinter den betreffenden Text.

Text läßt sich leider auch nicht grafisch schreiben. Deshalb kann es vorkommen, daß beim grafischen Schreiben (GOSUB 650) etwas vom darunter liegenden Bild verlorengeht. Der Hintergrund bleibt aber erhalten und wird bei CN = 1 rekonstruiert.

Die Tonausgabe (GOSUB 400) erfolgt über den TAPE-Ausgang des KC87 und den eingebauten Summer in einheitlicher Lautstärke. Ist SV = 0, hören Sie keinen Ton.

Wollen Sie einen Drucker anschließen, so benötigen Sie ein Erweiterungsmodul. Initialisieren Sie Ihren Drucker am besten vor dem Start von BASICODE.

Für den Einsatz von Speichererweiterungen gelten die gleichen Bedingungen wie unter Standard-BASIC.

Sind Sie Nutzer eines KC85/1 oder Z9001, ist "BAC87" nur lauffähig, wenn zusätzlich ein BASIC-Modul gesteckt ist. Es existieren jedoch BASIC-Module aus einer ersten Serie (1985), die über ein abweichendes Bit-Muster verfügen. Der Bascoder für den KC87 kann dort nicht eingesetzt werden. Bitte wenden Sie sich in einem solchen Fall über den Rundfunk der DDR oder über den Herausgeber an das Autorenkollektiv.

Der Bascoder ist vollständig in Maschinensprache geschrieben. Die Subroutinen der BASIC-Zeilen 10-950 enthalten ausschließlich Sprunganweisungen in das Maschinenprogramm, wo auch alle Parameterübergaben an den Interpreter erfolgen. Diese Sprunganweisungen dürfen vom Nutzer des Bascoders nicht verändert werden und sind deshalb für ihn nicht sichtbar. Die Subroutinen lassen sich im Direkt-Mode nur aufrufen, wenn der rufende Befehl oder eine geeignete Subroutine (GOSUB 100) an erster Stelle der Eingabezeile steht. Dazu ein Beispiel:

```
>GOSUB 25Ø                                        ein Signalton
>FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I                 UL ERROR
>GOSUB 1ØØ: FOR I=1 TO 1Ø: GOSUB 25Ø: NEXT I      zehn Signaltöne
```

5.4. Bascoder für den Commodore C 64

Laden und Starten des Bascoders

Nach dem Auslösen eines RESET bzw. dem Aus- und Wiedereinschalten des Computers (dies ist zu empfehlen, um Fehlreaktionen des Computers beim Start des Bascoders zu verhindern) kann der Bascoder durch Eingabe von:

```
LOAD"C-64 BASICODE-3"
```

oder einfach

```
LOAD
```

und Drücken der RETURN-Taste von Kassette geladen werden. Nach dem Erscheinen der Aufforderung

```
PRESS PLAY ON TAPE
```

schalten Sie die Datasette auf Wiedergabe. Wenn der Computer den Programmanfang gefunden hat, meldet er sich mit

```
FOUND C-64 BASICODE-3.
```

Nach ca. 10 Sekunden, bzw. dem Drücken der Leer- oder Commodore-Taste, beginnt das Einlesen des Bascoders. Dieser Vorgang ist nach ca. 2:40 min beendet, worauf der Computer die Datasette stoppt und sich mit

```
READY
```

meldet. Nun kann der Bascoder durch Eingabe des Kommandos "RUN" und Drücken der RETURN-Taste gestartet werden. Durch den Start des Bascoders laufen im wesentlichen folgende Operationen automatisch ab:

- Anzeige des Titelbildes, das den Programmnamen, Namen und Adresse des Programmautors sowie einen Hinweis, wie man zum Menü gelangt, enthält.
- Heruntersetzen des Zeigers für das Ende des Basic-RAM auf 8000H.
- Umschaltung auf den 2. Zeichensatz (Klein-/Großschrift-Modus) und Blockieren der Zeichensatzumschaltung.
- Setzen der Rahmen-, Hintergrund- und Cursorfarbe (Rahmen - grau1; Hintergrund - schwarz; Cursor - grau3).

Abschließend erscheint in der 1. Bildschirmzeile die Ausschrift

```
BASICODE - IN BETRIEB !
```

und die Meldung READY. Damit ist der Computer für die Arbeit im BASICODE-Modus bereit.

Beschreibung der Befehle

Durch Drücken der RETURN-Taste und nachfolgend der Funktionstaste F1 gelangt man zum Menü des Bascoders. Darin werden folgende Optionen zur Auswahl angeboten:

Taste	Option
F1	Abspeichern eines Programms im BASICODE-Format auf Kassette
F3	Laden eines BASICODE-Programms von Kassette und übersetzen ins Commodore C 64 BASIC
Leertaste	zurück zu BASIC

Wenn dieses Menü besteht, so bewirkt das Drücken jeder Taste, außer den o. g. und außer SHIFT, COMMODORE, RESTORE, CTRL
den Neuaufbau des Menü's.

F1 Abspeichern eines Programms im BASICODE-Format auf Kassette

Eine wichtige Voraussetzung für die Anwendung dieses Befehls besteht darin, daß sich ein fehlerfrei arbeitendes Programm im Speicher befindet. Nur dann kann erreicht werden, daß der Bascoder ein einwandfreies BASICODE-Programm auf die Kassette schreibt. Aus dem Menü - welches, falls noch notwendig, durch Drücken von RETURN und F1 erreicht wird - wird durch Drücken von F1 die SAVE-Routine aktiviert. Daraufhin führt der Computer als erstes eine Kontrolle des Programms

auf unzulässige BASIC-Befehle sowie die Einhaltung einer maximalen Zeilenlänge von 60 Zeichen durch. Dieser Vorgang geht sehr schnell vonstatten und läßt sich auf dem Bildschirm verfolgen. Wird dabei ein Fehler erkannt, stoppt der Computer mit der Meldung

```
KOMMANDO NICHT ERLAUBT ERROR IN <Zeilennummer>
```

bzw.

```
LINIE ZU LANG ERROR IN <Zeilennummer>.
```

Hierauf ist die fehlerhafte Zeile zu "listen" und zu korrigieren. Danach kann die SAVE-Routine, wie oben beschrieben, erneut aktiviert werden. Dieser Vorgang ist so lange zu wiederholen, bis das Programmende fehlerfrei erreicht ist. Dann meldet sich der Computer mit der Aufforderung:

```
PRESS RECORD & PLAY ON TAPE.
```

Jetzt wird die Datasette gemäß der Aufforderung gestartet, und es beginnt das Abspeichern des BASICODE-Programms auf Kassette. Dabei werden eventuell fehlende Anführungszeichen am Ende eines Strings, was im Commodore-BASIC zulässig ist, automatisch eingefügt.

Der Vorgang des Abspeicherns läßt sich auf dem Bildschirm über die Änderung der Farbe des Bildschirms verfolgen. Für die Dauer des Vortones ist er grün gefärbt, und während des Abspeicherns des eigentlichen Programmtextes wechselt die Farbe etwa jede Sekunde.

F3 Laden eines BASICODE-Programms von Kassette und Übersetzung ins Commodore C 64 BASIC

Die Kassette wird unter Nutzung des Zählwerkes an den Anfang des gewünschten BASICODE-Programms positioniert. Aus dem o. g. Menü wird durch Drücken von F3 die LOAD-Routine aufgerufen. Der Computer fordert Sie daraufhin mit der Meldung

```
PRESS PLAY ON TAPE
```

auf, die Datasette zu starten. Mit dem Start der Datasette färbt sich der Bildschirm rot. Der Computer sucht dann nach dem Vorton. Hat er ihn erkannt, nimmt der Bildschirm für die Dauer des Vortones eine grüne Farbe an. Beim nun folgenden Einlesen und Übersetzen des Programms ändert sich die Bildschirmfarbe, analog der SAVE-Routine, etwa jede Sekunde. Dieser periodische Farbwechsel ist ein Kennzeichen dafür, daß ordnungsgemäß eingelesen wird. Für den Nutzer zunächst unsichtbar, werden die eingelesenen und übersetzten Programmzeilen während des Einlesens in der 1. Bildschirmzeile angezeigt. Nur bei einem Abbruch oder am Ende der LOAD-Routine wird die zuletzt eingelesene Programmzeile sichtbar.

Ein vorzeitiger Abbruch der Routine kann durch gleichzeitiges Drücken von RUN/STOP und RESTORE oder das Abschalten der Datasette erzwungen werden.

Tritt beim Einlesen und Übersetzen ein Fehler auf, so meldet dies der Computer durch die Ausschrift

```
? LOAD ERROR.
```

Man kann das Programm jetzt listen, um eventuell so den Fehler zu finden und zu korrigieren. Oft ist es jedoch leichter, das Programm zu starten. Denn mit hoher Wahrscheinlichkeit wird es beim Erreichen eines Fehlers zum Absturz des Programms oder zu Fehlreaktionen des Computers kommen. Auf diese Weise lassen sich zumindest Syntax-Fehler relativ einfach lokalisieren.

Kann der Fehler nicht gefunden werden, oder wurde das Programm nur unvollständig eingelesen, so ist das Einlesen wie oben beschrieben zu wiederholen.

Leertaste zurück zu BASIC

Durch Drücken der Leertaste wird das Menü verlassen, und der Computer ist für die Arbeit in BASIC bereit.

Besonderheiten des Bascoders

Der Bascoder für den C 64 wurde von M. v. Deelen aus den Niederlanden geschrieben. Er besteht aus einem ca. 6,7 kByte langen Programmteil in Maschinensprache für die BASICODE-Subroutinen und einem 667 Byte langen BASIC-Programm für das Titelbild. Da das Maschinenprogramm auf den RAM oberhalb Adresse 8000H zugreift, ist der RAM ab dieser Adresse vor Überschreiben durch Basic-Programme geschützt. (Dies erfolgt automatisch beim Start, s. o.) Damit stehen nach dem Start des Bascoders noch 33782 Byte für die Arbeit mit BASICODE zur Verfügung.

Die BASICODE-Subroutinen mit Zeilennummern kleiner 1000 existieren bei diesem Bascoder physisch nicht. Statt dessen sind sie als zusätzliche BASIC-Befehle vorhanden, d.h., sie stellen eine Erweiterung des im C 64 vorhandenen BASIC-Interpreters dar. Dadurch steht gegenüber anderen Bascodern für den C 64 ca. 1 kByte mehr Speicherplatz für Basic-Programme zur Verfügung. Außerdem ist es dadurch möglich, die BASICODE-Subroutinen, wie die Mehrzahl der BASIC-Befehle des C 64, für eine Eingabe im Direktmodus zu verwenden.

Bei der Arbeit mit einer Diskettenstation (Floppy) empfiehlt es sich, nach dem Start des Bascoders den NEW-Befehl einzugeben, um zu vermeiden, daß der Bascoder beim SAVEn eines Programms mit abgespeichert wird.

Der Bascoder kann wie ein gewöhnliches BASIC-Programm kopiert werden, indem man ihn nach dem Laden (es darf dazu auf keinen Fall gestartet worden sein) auf Kassette mit

```
SAVE"Ø: Programmname"
```

oder auf Diskette mit

```
SAVE"Ø: Programmname",8
```

abspeichert. Hierzu kann man auch ein beliebiges Kopierprogramm benutzen.

Der Textschirm besteht aus 25 Zeilen zu je 40 Zeichen, wobei sowohl Groß- als auch Kleinbuchstaben zugelassen sind.

Bei der Verwendung der Subroutinen GOSUB 200, 210 und 220 wird bei Betätigen der CTRL-, RESTORE-, SHIFT- oder COMMODORE-Taste an die Variable IN$ kein Zeichen und an IN kein Tastaturcode abgegeben. Weiter sind alle Tastenkombinationen, welche die CTRL- bzw. COMMODORE-Taste benutzen, zu vermeiden (z. B. Farb- und Funktionstasten). Sie übergeben nicht an die Variable IN den Tastaturcode und auch nicht einen ASCII-Code an IN$, der dem erzeugten Zeichen entspricht. Sonstige abweichende Code sind in nachfolgender Tabelle aufgeführt.

Taste		IN$	IN	
HOME		*	-20	
CLR		*	-20	
INST		*	-21	
PI		gra.	94	
STOP		*	-4	
SHIFT und	+	gra.	91	
	-	gra.	93	
	*	gra.	64	
	@	Blockhaken	58	
		gra.	41	gra. - Grafiksymbol

Für hochauflösende Grafik stehen horizontal 288 Pixel und vertikal 200 Pixel zur Verfügung.

Bei der Eingabe eines falschen Kommandos oder beim Auftreten eines Fehlers während des Programmlaufes wird ein Piep-Ton und eine entsprechende Fehlermeldung erzeugt.

Bei der Tonerzeugung wird für SP = 0 der tiefste Ton und bei SP = 106 die größtmögliche Tonhöhe erreicht.

Der C 64 ist in der Lage, mit maximal 3 Beständen gleichzeitig zu arbeiten, und zwar mit einem auf Kassette im BASICODE- oder Commodore-Format (NF = 0,1 bzw. 2,3) und mit zwei auf Diskette (NF = 4...7).

Enthält NF$ beim Aufruf der Subroutine GOSUB 500 mehr als 7 Zeichen, so werden nur die ersten 7 Zeichen verwendet, der Rest wird ignoriert. Die Routine fügt automatisch vor diesem String ein "b" ein. Wird z. B. ein Bestand durch `Zeilennummer NF=5: NF$="test": GOSUB 500` auf Diskette eröffnet, so wird er in der Directory als sequentielles File mit dem Namen "btest" eingetragen.

Auf der Diskette bereits existierende Bestände können nicht durch BASICODE überschrieben werden. Sollte es trotzdem versucht werden, z. B. durch Eröffnen eines Bestandes mit schon existierendem Filenamen, wird in der Variablen IN der Wert -1 zurückgeliefert.

Ebenso ist es unzulässig, gleichzeitig 2 Bestände gleichen Namens auf einer Diskette anzulegen. In diesem Falle werden 2 Files gleichen Namens, aber unterschiedlichen Filetyps angelegt, nämlich ein DELeted und ein SEQuentielles File. Beim Versuch, Daten in diesen Files mittels GOSUB 560 abzulegen, wird in IN ebenfalls -1 zurückgeliefert.

Das Einlesen bzw. Abspeichern von Beständen im BASICODE-Format erfolgt ähnlich dem bereits beschriebenen Einlesen bzw. Abspeichern von Programmen im BASICODE-Format, jedoch erscheinen die Aufforderungen zur Bedienung der Datasette hierbei in der untersten Bildschirmzeile.

Der C 64 weicht deutlich vom ASCII-Code ab. Es kommt bei einigen BASICODE-Programmen zu Problemen. Nicht alle Autoren von BASICODE-Programmen kennen dieses Problem ausreichend gut. Deshalb ist es nützlich, wenn C 64-Besitzer die Programmautoren oder das Autorenkollektiv dieses Buches auf solche Mängel aufmerksam machen.

5.5. Bascoder für Commodore C plus/4 und C-16 (64k)

Dieser Bascoder stand erst ganz kurz vor Abschluß des Buches zur Verfügung. Wegen der großen Anzahl entsprechender Rechner wurde er, wenn auch mit relativ kurzer Beschreibung, sofort einbezogen. Er liegt als Turbo-File vor und ist nur in einer der beiden Phasenlagen einlesbar. Er wird geladen gemäß:

```
LOAD" <RETURN>
```

Anschließend kann der Bascoder (bevor er mit RUN gestartet wurde) mittels

```
SAVE"BASCODER C+/4)"
```

auf Kassette oder auf Diskette mittels

```
DSAVE"BASCODER C+/4)"
```

kopiert werden.

Soll die Kopie als Turbo-File erfolgen, muß zuvor das entsprechende TURBOSAVE-Programm in den Rechner eingelesen werden.

Bedienungshinweise

Nach dem Laden wird der Bascoder mit <RUN> gestartet. Nach einer kurzen Pause erscheint einmalig das Titelbild mit kurzen Hinweisen zur Bedienung. Nach dem Betätigen der <SPACE>-Taste befinden sich die Subroutinen im Speicher des Rechners, und ein BASICODE-Programm kann mit <F2> + <RETURN> geladen werden. Das Programm ändert die Belegung folgender Funktionstasten:

F1 = sys5056:rem menu	Aufruf des Menüs
F2 = sys4955:rem LOAD basicode	Laden von BASICODE-Programmen
F4 = sys5421:rem SUBROUTINES	Bereitstellung der Subroutinen
F5 = sys4629:rem SAVE basicode	Sichern von BASICODE-Programmen

Menüfunktionen

*** F1**
Nach Aufruf von <F1> + <RETURN> erscheint ein Menü mit der Belegung der Funktionstasten F1, F2,F4, F5, der Versionsnummer und dem Hinweis, kein GRAPHICCLR zu benutzen.

*** F2**
Mit <F2> + <RETURN> wird ein BASICODE-3-Programm von Kassette geladen und an das momentan im Speicher stehende Programm angehängt.
Hinweis: Vor dem Laden eines neuen Programms soll immer <F4> + <RETURN> aufgerufen werden, um den Speicher ab Zeile 1000 frei zu machen.

*** F4**
<F4> + <RETURN> stellt die BASICODE-3-Subroutinen von Zeile 10 bis 999 bereit. Gleichzeitig wird der BASIC-Speicher geleert.

*** F5**
Die Tastenkombination <F5> + <RETURN> speichert ein BASICODE-File auf Kassette. Dabei erfolgt eine Abfrage nach Anfangs- und Endzeile. Werden diese mit <RETURN> beantwortet, so wird der Anfang auf 1000 gesetzt und die Endzeile automatisch ermittelt. Durch Eingabe von Zeilenwerten ist es möglich, auch einen Teil des Programms zu speichern.

Besonderheiten des Bascoders

Bildschirmformat	40*25	Zeichen	HO = 39	VE = 24
Grafikmodus	250*200	Pixel	HG = 250	VG = 200
Textzeichen im Grafik-Mode	31*25	Zeichen zu 8*8 Pixel		
Rechengenauigkeit	9	Ziffern		
Grenze für SP	45 bis 127			

Die ASCII- und Steuerzeichen entsprechen denen des Commodore C-64.
Wird ein nicht vorhandenes Diskettenlaufwerk angesprochen, so erscheint:

```
"FILE OPEN ERROR"
```

und nicht

```
"?DEVICE NOT PRESENT ERRROR"
```

Während des Lesens von bzw. Schreibens auf Kassette bleibt der Bildschirm leer. Es erfolgt auch kein Farbwechsel wie beim C-64.

Das Laden eines Files kann nur mit <RUN/STOP>/RESET unterbrochen werden. Über

<X> + <RETURN> wird der Maschinensprachemonitor verlassen, und der Computer geht in den Direktmodus über. MIT <RUN> + <RETURN> wird erneut der Bascoder gestartet.

Wenn in der Subroutine 280 ein BREAK erlaubt ist, dann wird durch Betätigen der Taste <RUN/STOP> ein Programm über GOTO 950 beendet und gibt eine BREAK-Meldung mit Zeilennummer aus.

Der Grafikbefehl GRAPHIC CLR darf in BASICODE nicht benutzt werden. Es erfolgt ein Rechnerabsturz. Beim C-16 mit 16 kByte Speicher erfolgt eine Blockierung, bei 32 kByte Speicher erfolgt die Fehlermeldung:

```
"kein 64k c-16 o. Plus/4 64k Speicher".
```

5.6. Bascoder für den Z1013

Rechnerseitige Voraussetzungen

Der Mikrorechnerbausatz Z1013 besitzt keinen internen BASIC-Interpreter. Nach dem Einschalten meldet er sich mit seinem maschinencode-orientierten Betriebssystem. Der BASIC-Interpreter muß erst von der Kassette eingelesen werden. Danach stehen bei 16 kByte RAM noch ca. 5 kByte Speicher für Programme zur Verfügung. Dieser Platz reicht nicht für die Nutzung eines gesonderten Bascoders aus. BASICODE wäre also für den Z1013 in seiner 16k-Variante unerreichbar. Entgegen dem eigentlichen BASICODE-Konzept wurde deshalb ein eigenständiger BASIC-Interpreter entwickelt, welcher der Syntax und den Möglichkeiten des BASICODEs weitestgehend entspricht. Sie werden bereits beim Programmieren auf viele zu berücksichtigende Besonderheiten hingewiesen. Ein mit dem Z1013-Bascoder lauffähiges BASIC- Programm entspricht mit hoher Wahrscheinlichkeit den BASICODE-Normen.

Der Mikrorechnerbausatz Z1013 existiert in mehreren industriell hergestellten Versionen, die sich nicht nur im vorhandenen freien Arbeitsspeicher, sondern auch in der Taktfrequenz und im Betriebssystem, dem sogenannten Monitor, unterscheiden. Zusätzlich bestehen verschiedene Tastaturanpassungen. Der Bascoder des Z1013 versucht, möglichst vielen Versionen gerecht zu werden. Ihr Rechner möchte jedoch folgenden Anforderungen genügen:

- Der Rechner arbeitet mit einer Taktfrequenz von 2 MHz.
- Er besitzt eines der industriell hergestellten Betriebssysteme 2.02 oder A.2, bzw. dieses ist mit ihnen kompatibel. Das ist garantiert der Fall, wenn an ihrem Rechner noch keine Veränderungen vorgenommen wurden.
- Ihr Z1013 besitzt nach dem Einschalten mindestens 16 kByte freien Arbeitsspeicher, in den dann der Bascoder eingelesen werden kann. Für die Nutzung des 64k-Bascoders benötigen Sie die 64k-Variante des Z1013.

Für die Anwendung von BASICODE wurden zwei Varianten als Bascoder entwickelt:

- Der große Bascoder verlangt eine Speicherkapazität von 64 kByte. Er erfüllt alle Forderungen von BASICODE, und die Arbeit ist mit ihm weitgehend automatisiert.
- Der kleine Bascoder benötigt nur 16 kByte und besteht aus zwei getrennt zu ladenden Teilen: BASCON und dem eigentlichen kleinen Bascoder. Daher ist für das vollständige BASICODE-Protokoll eine aufwendigere manuelle Arbeit erforderlich. Andererseits sind mit den beiden Teilprogrammen einige Sonderfunktionen möglich, die beim Arbeiten mit dem großen Bascoder zur Vereinfachung der Nutzung automatisiert übersprungen werden.

Das Konvertierungsprogramm BASCON hat die Aufgabe, die BASICODE-Textfiles vom BASICODE-Übertragungsformat in das KC-Format zu übersetzen und umgedreht. Das KC-Format entspricht etwa dem Aufzeichnungsverfahren, welches durch das BASIC-Kommando CSAVE realisiert wird. Das bedeutet, daß jedes BASICODE-Programm zunächst mit BASCON in einen Zwischencode gewandelt und danach auf eine Kassette abzuspeichern ist. Von dort kann der Bascoder das BASICODE-Programm direkt in die interne Codierung übersetzen, wie dieses durch größere Rechner mit Diskettenbetrieb geschieht (Abb. 5.13). Analog geschieht die Übersetzung eines BASICODE-Programms von der internen Darstellung auf Kassette. Es muß dann durch BASCON noch in das austauschbare BASICODE-Format konvertiert werden. Dieser etwas umständliche Kompromiß bleibt erspart, wenn die 64k-Variante des Bascoders genutzt werden kann.

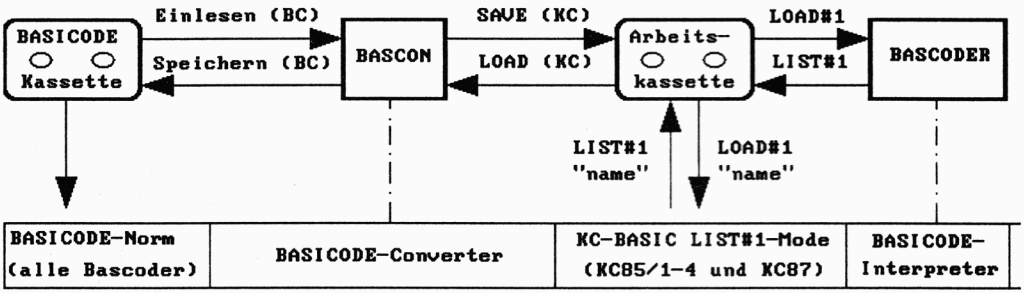


Abbildung 5.16. Kassettenschema

Einlesen und Bedienen der 64k-Variante des Bascoders

Zum Einlesen der 64k-Variante des Z1013-Bascoders stellen Sie bitte folgendes Kommando auf dem Bildschirm bereit:

```
# L 1ØØ 36FF
```

Nun legen Sie die Kassette mit dem 64k-Bascoder in den Kassettenrecorder, spulen zum Anfang des Programms vor und starten die Kassette mit Wiedergabe. Alle Z1013-Programme wurden mit *maintape* aufgezeichnet und besitzen einen Kopfblock, in dem die jeweiligen Programmparameter abgespeichert sind. Beide Teile lassen sich deutlich durch einen zweiten Vorton unterscheiden. Bestätigen Sie deshalb Ihr obiges Kommando erst beim Ertönen des zweiten Vortones mit der ENTER-Taste. Besitzen Sie in Ihrem Programmbestand Headersave oder *maintape*, dann lassen sich natürlich auch diese Routinen zum Einlesen nutzen. Beim *maintape* haben Sie zusätzlich den Nachweis, daß die CRC-Prüfsumme übereinstimmt, d. h., es ist mit sehr großer Sicherheit kein Lesefehler aufgetreten.
Das fehlerfrei eingelesene Programm starten Sie mit:

```
# J 3ØØ   ENTER .
```

Der 64k-Bascoder meldet sich wie folgt:

```
Welcome by BASICODE-3  64kB
Memory-Size: _
```

Der Bascoder erwartet die dezimale Eingabe des für ihn zulässigen höchsten Speicherplatzes. Hier genügt die Betätigung der ENTER-Taste, und der Bascoder meldet sich mit:

```
32767 Bytes free
```

```
OK
>
```

Zusätzlich wurde das Programm *maintape* ab Speicheradresse 0E000H umgelagert. Es ermöglicht die Nutzung der inclusiven Kommandos HSAVE und HLOAD, womit die erstellten BASICODE-Programme im rechnereigenen Kassettenaufzeichnungsverfahren rationell abgespeichert, eingelesen und kopiert werden können. Der durch das Umlagern freigewordene Platz hinter dem 64k-Bascoder wird für Arbeitszellen und das BASICODE-Programm genutzt. Vervielfältigen Sie deshalb den 64k-Bascoder immer, ohne ihn vorher zu starten. Er wird sonst nicht in seiner vollständigen Form abgespeichert.

Weitere Programmteile, z. B. Druckerroutinen, sind ab 0D000H bis 0DFFFH bzw. ab 0E500H in den Speicher einzutragen und werden durch den 64k-Bascoder nicht überschrieben.

Einlesen und Speichern von BASICODE-Programmen mit dem 64k-Bascoder

Im 64k-Bascoder wurden alle Anforderungen des BASICODE-3 realisiert. BASICODE-Programme lassen sich mit dem Kommando

```
BCLOAD
```

einlesen. Zum Speichern dient das Kommando

```
BCSAVE
```

Achten Sie bitte auf die unterste Zeile des Bildschirmes. Hier wird aufgefordert, das Kassettengerät auf Wiedergabe bzw. Aufnahme zu stellen und zu starten. Alle Meldungen in dieser Zeile sind mit ENTER zu bestätigen. Damit wird der ursprüngliche Zeileninhalt rekonstruiert und mit der Ausführung bzw. Fortsetzung des Programms begonnen.

Durch das Kommando BCLOAD werden in den untersten acht Bildschirmzeilen alle erkannten Zeichen dargestellt, ohne diese bereits einzulesen. Sie können sich also auf Ihrer Kassette orientieren und anhand der jetzt sichtbaren Zeilennummern und Programmzeilen das gewünschte BASICODE-Programm durch schnelles Vor- und Zurückspulen mit dem Kassettengerät heraussuchen. Erkennt der Bascoder einen Vorton, dann löscht er die untersten acht Bildschirmzeilen und meldet sich mit der Information:

```
Ich lese ein!
```

Gleichzeitig synchronisiert er seine Zeitkonstanten für das Einlesen auf das betreffende File und vermindert so die Anzahl der Lesefehler. Das Einlesen können Sie, wie das Suchen, im unteren Bildschirmviertel mitverfolgen. Auch während des sich anschließenden Übersetzungsvorganges werden alle Programmzeilen nochmals auf dem Bildschirm dargestellt. In der Regel können Sie nun mit RUN die Abarbeitung starten. Auch GOTO 1000 ist möglich. Autostart ist nicht vorgesehen.

Nach Lesefehlern ist das Einlesen zu wiederholen oder von den Möglichkeiten der integrierten Funktion "Textkorrektur" Gebrauch zu machen, wie dieses beim Programm BASCON (kleiner Bascoder) beschrieben ist.

Einlesen von BASCON

Wenn Sie nur die 16 kByte-Variante des Z1013 besitzen, dann müssen BASICODE-Programme zuerst mit dem Konvertierungsprogramm BASCON in einen Zwischencode verwandelt werden. Das Programm BASCON eignet sich außerdem zum schnellen Kopieren von BASICODE- Programmen. Zum Einlesen von BASCON ist das folgende Kommando auf dem Bildschirm bereitzustellen:

```
# L 1ØØ EFF
```

Nun verfahren Sie, wie unter dem 64k-Bascoder beschrieben. BASCON läßt sich mit dem Kommando

```
# J 1ØØ ENTER
```

starten.

Beschreibung von BASCON

Nach dem Start prüft BASCON seinen verfügbaren Arbeitsspeicher und trägt dessen Grenzen in seine Arbeitszellen ein. Dann meldet er sich mit seinem Menü-Bild.

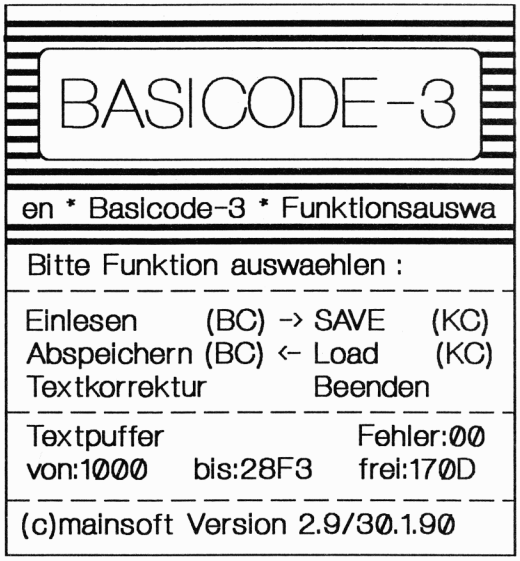


Abbildung 5.17. Startbild Z1013

Ein Kommando wird ausgewählt, indem der jeweilige Anfangsbuchstabe auf der Tastatur gedrückt wird. Jetzt blinkt das entsprechende Kommando auf dem Bildschirm und der Rechner wartet auf Ihre Bestätigung mit der ENTER-Taste. Eine Korrektur ist möglich, indem Sie einen anderen Anfangsbuchstaben eingeben. Jeweils nach Ausführung eines Kommandos erwartet BASCON eine Tastaturbetätigung. Damit werden die Kommandonachrichten gelöscht und ins Menü-Bild zurückgekehrt. Alle Kommandoausführungen lassen sich durch Betätigung einer Taste unterbrechen. Es erscheint die Nachricht:

```
TASTATUR-BREAK!
```

Bei der Original-Flachfolientastatur sind die Tasten der vierten Spalte wirksam (S4/K). Anpassungen an andere Tastaturen sind möglich. Die Textkorrektur wird mit der BREAK-Taste (S4/K bzw. ^C) vorzeitig beendet.

Im Menü stehen folgende Kommandos zur Verfügung:

Einlesen (BC) BASICODE-Programme oder -Bestände im BASICODE-Format werden von der Kassette in den Rechner eingelesen.

Save (KC) Das in den Rechner eingelesene Programm oder der Bestand wird im Arbeitsformat des Bascoder's (KC-Format) auf Kassette abgespeichert und kann durch den Bascoder von dort in den Rechner eingelesen werden.

Load	(KC)	BASICODE-Programme oder -Bestände werden im Arbeitsformat des Bascoders von der Kassette in den Rechner eingelesen.
Abspeichern	(BC)	Das in den Rechner eingelesene Programm oder der Bestand wird im BASICODE-Format auf Kassette abgespeichert.
Textkorrektur		Mit Hilfe des in BASCON integrierten einfachen Editors können Fehler beim Einlesen der BASICODE-Programme oder -Bestände im eingeschränkten Maße korrigiert werden.
Beenden		Es wird zum Monitor zurückgesprungen.

Im unteren Teil des Menübildes erhalten Sie Auskunft über Beginn und Ende des Textpuffers, den noch freien Speicherbereich und die Anzahl der beim Einlesen des BASICODE-Textfiles aufgetretenen Fehler. Dabei werden die Zeichen gezählt, die einen ASCII-Code kleiner als 32 oder größer als 126 haben und demzufolge mit "#" überschrieben werden. Die Prüfung auf Fehler und das Überschreiben mit "#" erfolgen jeweils nur nach dem Kommando "Einlesen BC". Alle Zahlenangaben erfolgen der Einfachheit halber hexadezimal.

Beschreibung der BASCON-Kommandos

Einlesen

Nach Ansprung des Kommandos werden Sie aufgefordert, das Kassettengerät mit Wiedergabe zu starten. Bestätigen Sie die Ausführung mit einer Tastaturbetätigung. BASCON zeigt Ihnen daraufhin alle interpretierbaren Zeichen im untersten Bildschirmviertel an. Erkennt BASCON den Beginn eines BASICODE-Textfiles, so meldet er sich mit der Ausschrift:

```
Ich lese ein !
```

Gleichzeitig synchronisiert er seine Zeitkonstanten mit dem einzulesenden File, um dadurch die Anzahl der Lesefehler weiter zu senken. Kann BASCON danach kein BASICODE-Textfile erkennen, so bricht der Suchvorgang ab und meldet sich mit :

```
Zeitfehler
```

Der eingelesene Text wird über acht Zeilen in der unteren Bildschirmhälfte dargestellt. Man kann sich so einen Überblick über den Inhalt verschaffen.

Ist der verfügbare Speicherplatz zur Ablage des Textfiles nicht ausreichend, wird der Einlesevorgang unterbrochen und

```
Pufferüberlauf
```

gemeldet. Das bis zu diesem Zeitpunkt eingelesene Textfile ist trotzdem im Textpuffer verfügbar und kann in seiner unvollständigen Form weiterverwendet werden. Diese Situation wird aber aufgrund der gewählten Arbeitstechnologie auch bei 16 kByte Speicherplatz kaum auftreten.

Mit normalem Ende des Lesevorgangs werden das Prüfbyte kontrolliert und in Abhängigkeit vom Ergebnis ausgeben:

```
***LESEFEHLER***
```

```
PROGRAMM OK
```

oder

```
DATEI OK
```

Stimmt das Prüfbyte mit dem Kontrollwert überein und Sie werden trotzdem im Menübild über Fehler informiert, dann enthielt das Textfile bereits beim Abspeichern unzulässige Zeichen.

In Abhängigkeit vom Prüfergebnis werden Ihnen im Menübild die Kommandos "Textkorrektur" oder "Save (KC)" angeboten.

Wollen Sie BASICODE-Textfiles nur vervielfältigen, dann wählen Sie bitte "Abspeichern (BC)".

Save (KC)

Nach Aufruf dieses Kommandos erscheint die Aufforderung:

```
NAME :
```

Hier geben Sie bitte einen maximal achtstelligen Namen an, mit dem sie das im Puffer vorhandene Textfile auf Kassette abspeichern und bei Notwendigkeit wieder einlesen können. So aufgezeichnete Programme sind für die Eingabe in einen Z1013-Bascoder bestimmt. Sie lassen sich aber auch mit LOAD#1"name" in die KC-Rechner einlesen und über den dortigen Bascoder abarbeiten.

Starten Sie Ihr auf Aufnahme gestelltes Kassettengerät, wenn Sie die Namenseingabe mit ENTER beenden, da keine nochmalige Abfrage erfolgt. Das Saven des Textfiles läßt sich am hexadezimalen Blockpointer verfolgen. Ist es beendet, erscheint die Ausschrift:

```
VERGLEICH?
```

Geben sie ein "J" ein, dann können Sie die richtige Aufzeichnung anhand der Blockprüfsummen kontrollieren lassen. Es erscheint:

```
ZURUECK!
```

Stellen Sie das Kassettengerät auf den Beginn Ihrer Aufzeichnung und betätigen Sie ENTER. Bei Lesefehlern wird

```
FEHLER!
```

ausgegeben. Nach ENTER ließe sich der Leseversuch wiederholen. S4/K bzw. ^C bricht ihn ab.

Abspeichern (BC)

Nach Aufruf des Kommandos werden Sie aufgefordert, Ihr Kassettengerät auf Aufnahme zu stellen und zu starten. Auch hier wartet BASCON auf die Bestätigung mit Tastendruck. Die entstehenden Kassettenfiles lassen sich mit jedem anderen Bascoder einlesen. Fehlt das Textfile, so wird eine Fehlermeldung:

```
TEXTPUFFER LEER !
```

ausgegeben.

Load (KC)

Nach diesem Kommandoaufruf erwartet BASCON die Eingabe eines Namens. Stimmt dieser nicht mit dem Namen des anliegenden Kassettenfiles überein, wird das Einlesen mit

```
FEHLER!
```

abgewiesen. Eingelesen werden können die von den Z1013-Bascodern und auch die von den KC-Rechnern mit LIST#1"name" abgespeicherten Programme. BASCON unterscheidet selbständig zwischen BASICODE-Programmen und -Beständen, die vom Z1013-Bascoder aufgezeichnet wurden. Entprechend erscheinen die Meldungen:

```
PROGRAMM OK
```

oder

```
BESTAND OK .
```

Bei Lesefehlern

```
BAD RECORD
```

oder

```
RECORD NOT FOUND
```

spulen Sie bitte Ihre Kassette etwas zurück, betätigen die ENTER-Taste und versuchen, den fehlerhaften Block erneut zu lesen. Abbrechen lassen sich die Leseversuche mit der BREAK-Taste (S4/K bzw. ^C).

Textkorrektur

Dieses Kommando ist notwendig, da u. a. fehlerhafte Zeilennummern zu Problemen beim Einlesen in den Bascoder führen. Beim Aufruf dieses Kommandos sucht BASCON alle Programmzeilen heraus, die fehlerbehaftet sein können, und bietet sie nacheinander zur Korrektur an.
Seine Auswahlkriterien sind:

- Die erste Zeilennummer ist nicht 1000.
- Der Zeilenabstand erfolgt nicht in aufsteigender Reihenfolge und in Zehnerschritten.
- Hinter der Zeilennummer folgt kein Leerzeichen.
- Das erste Zeichen der Zeile nach diesem Leerzeichen ist kein Großbuchstabe.
- Die Zeilenlänge ist ohne Leerzeichen größer als 60 Zeichen.
- Im Zeilentext existieren unzulässige Grafik- oder Steuerzeichen.

Stellt BASCON solche möglichen Fehler fest, dann erscheinen auf dem Bildschirm:

- die vorangegangene Zeile (wenn vorhanden),
- die wahrscheinlich fehlerbehaftete Zeile zur Korrektur,
- die nachfolgende Zeile (wenn vorhanden).

Dabei kann es durchaus vorkommen, daß z. B. ein von Zehn abweichender Zeilenabstand gerechtfertigt ist. Korrigieren Sie bitte die festgestellten Fehler durch Überschreiben. Ein Insert- oder Delete-Modus ist nicht vorgesehen, zumal der Bascoder genügend Editierkomfort bietet. Der Cursor läßt sich nicht über die Zeilengrenzen hinaus bewegen.

Mit Betätigung der ENTER-Taste werden die Zeilen vom Bildschirm in den Textpuffer übernommen. Dann erscheint die nächste fehlerhafte Zeile. Ein vorzeitiger Kommandoabbruch ist mit der BREAK-Taste (S4/K bzw. ^C) möglich. Zeilen werden geteilt, indem in der Zeile vor die nächste Zeilennummer ein Doppelpunkt ":" gesetzt wird. Dieser wird beim Rückspeichern in den Textpuffer als Zeilentrennung interpretiert.

Natürlich können auf diese Weise nicht alle Fehler erkannt werden. Der große Vorteil dieses Verfahrens besteht jedoch darin, daß Sie große Unterstützung bei solchen Fehlern finden, die zu Übersetzungsproblemen im Bascoder führen können.

Beenden

Das große Bildfenster wird eingestellt, der Bildschirm gelöscht und zur Monitoreingabe zurückgekehrt.

Warmstart von BASCON

Ein Warmstart des BASCON ist mit folgendem Kommando möglich:

```
# J 1Ø2 >ENTER< .
```

In diesem Fall werden die Arbeitszellen nicht initialisiert, und Sie können mit dem bereits im Speicher befindlichen Text weiterarbeiten. Auf diese Weise ist es auch möglich, irgendein Textfile im BASICODE- oder im KC-Format abzuspeichern. Dazu tragen Sie vor dem Ansprung der Warmstartadresse auf 0F10H und 0F11H das Ende Ihres eingelesenen Textes ein, der ab 1000H beginnen muß. Beachten Sie, daß zuerst das minderwertige, dann das höherwertige Byte der Endadresse vermerkt wird.

Einlesen des kleinen Bascoders

Den 16k-Bascoder laden Sie ähnlich wie den 64k-Bascoder in den Z1013 ein. Stellen Sie dazu bitte folgendes Kommando auf dem Bildschirm bereit :

```
# L 1ØØ 2AFF
```

Auch hier betätigen Sie die ENTER-Taste erst beim 2. Vorton. Der Start des Bascoders ist ebenfalls wie folgt möglich :

```
# J 3ØØ >ENTER<
```

Daraufhin erscheint die Abfrage :

```
Welcome to BASICODE-3  16kB

Memory-Size:
```

Hier kann eine Begrenzung des höchsten vom Bascoder genutzten Speicherplatzes erfolgen. In der Regel genügt eine Bestätigung mit ENTER ohne Zahlenangabe. In diesem Fall richtet sich der Bascoder seinen Platz entsprechend den Möglichkeiten des Rechners ein. Für Z1013-Varianten mit 64 kByte Speicherplatz wurde der höchste Speicherplatz auf 44040 (entspricht 0AC08H) begrenzt. Damit stehen 32767 Byte als Programmspeicher zur Verfügung. Alle darüber befindlichen Speicherplätze können vom Anwender ohne Einschränkung, z. B. für Druckertreiber, genutzt werden. Sinnvoller ist in diesem Fall jedoch die Verwendung des Bascoders für 64 kByte Speicher.

Der Bascoder meldet die ihm zur Verfügung stehende Anzahl freier Bytes.

Einlesen der BASICODE-Textfiles in den 16k-Bascoder

Wie bereits beschrieben, benötigt der 16k-Bascoder ein spezielles Arbeitsformat der Kassettenaufzeichnungen, welches Sie ihm mit BASCON zur Verfügung stellen können. Die so konvertierten BASICODE-Programme lassen sich wie folgt einlesen:

```
 OK
>LOAD#1"name" >ENTER<
>
```

Als Namen geben Sie den bereits beim Abspeichern mit BASCON gewählten an. Alle eingelesenen Programmzeilen werden Ihnen nacheinander auf dem Bildschirm aufgelistet und von dort selbständig in den Programmspeicher eingegeben. Folgende Fehlermeldungen sind möglich:

```
BAD FILE
```

und

```
FILE NOT FOUND
```

Sie können das Einlesen nach kurzem Zurückspulen der Kassette und ENTER fortsetzen.

```
SN ERROR
```

Es trat eine fehlerhafte Zeilennummer auf. Sie können versuchen, diese nach dem nochmaligen Einlesen dieses Arbeitsfiles mit "Load (KC)" in BASCON zu korrigieren. Eine Fehlerbearbeitung innerhalb des Bascoders ist nicht möglich, da die Eingabe des Arbeitsfiles an dieser Stelle unbedingt abgebrochen wird.

```
OM ERROR
```

Der zur Verfügung stehende Speicherplatz ist voll. Das eingelesene Programm ist nutzbar, wenn der bereits im Speicher vorhandene Teil, wenn auch nur eingeschränkt, eine Programmabarbeitung möglich macht. Eventuell lassen sich einige REM-Zeilen löschen.

Natürlich müssen Sie nicht jedes Mal beide Stufen der Übersetzung vom BASICODE-Textformat aus durchlaufen. Legen Sie sich ein Arbeitsband mit Ihren BASICODE-Programmen und -Beständen in der Arbeitsform Ihres Bascoders an, dann sinkt der Aufwand beträchtlich und Sie haben durch den zugeordneten Namen auch mehr Übersicht.

Durch die gewählte Form der Einspeicherung in den Bascoder lassen sich Programme auch untereinander mischen. Alle Programmzeilen werden entsprechend ihrer Zeilennummer in das Programm einsortiert. Mit einem vorherigen RENUMBER-Befehl können die bereits im Speicher befindlichen Zeilen entsprechend groß bzw. klein numeriert werden, um weitere Programme davor oder dahinter anzuketten.

Arbeit mit BASICODE-3-Beständen

Der 64k-Bascoder realisiert die Arbeit mit Beständen für NF = 0 bis NF = 3. Bei Angabe anderer Werte für NF (*Diskettenarbeit*) erfolgt die Meldung

```
I/O-ERROR
```

und der Sprung zum Direktmode. Beim 16k-Bascoder wurden diese Fehlerprüfungen weggelassen. Unabhängig vom konkreten Wert der Variablen NF arbeitet er mit NF = 2 für "Lesen von Beständen" und mit NF = 3 für "Schreiben von Beständen". Die Konvertierung in das BASICODE-Format, also NF = 0 oder NF = 1, muß hier über BASCON erfolgen.

Bei beiden Bascodervarianten darf jeweils nur ein BASICODE- Bestand eröffnet sein. Wird ein Programm mit der BREAK-Taste unterbrochen, in dem ein Bestand zur Ausgabe eröffnet war, dann erfolgen dessen Abschluß und die Ausgabe auf Kassette. Gleiches passiert beim Warmstart des Bascoders, wenn er wegen Speicherfehler oder ähnlichem abgestürzt war. Die so noch geretteten Daten lassen sich in einem folgenden Programmlauf wieder verwenden. Es soll gesichert werden, daß möglichst keine Daten verlorengehen.

Warmstart der Bascoder

Die Bascoder lassen sich vom Z1013-Monitor aus mit

```
# J 3Ø2 >ENTER<
```

restarten. Das bedeutet, daß ein eventuell schon im Speicher befindliches Programm für den Bascoder erhalten bleibt. Bedingung ist, daß die Arbeitszellen bereits geladen sind. Es ist also nicht möglich, einen Warmstart durchzuführen, ohne daß diesem bereits ein Start auf Adresse 300H vorausgegangen war.

Editierkommandos der Z1013-Bascoder

Der Kommandovorrat der Z1013-Bascoder beschränkt sich im wesentlichen auf die von BASICODE-3 vorgegebenen Befehle. Trotzdem ist es unumgänglich, ihn für die Programmerarbeitung und das Testen um einige Befehle zu erweitern :

NEW	RUN	STOP	RENUMBER
DELETE	EDIT	TRON	TROFF
AUTO	LINES	CSAVE	CLOAD
BYE	LIST	CONT	HLOAD
HSAVE	BCLOAD	BCSAVE	

Die mit CSAVE abgespeicherten BASICODE-3-Programme lassen sich nicht durch die KC-Rechner einlesen, da sich die internen Befehlsdarstellungen unterscheiden. Ein direkter Programmaustausch mit diesen Rechnern ist nur über LIST#1"name" bzw. LOAD#1"name" möglich.

Die Kommandos HSAVE und HLOAD des 16k-Bascoders speichern und lesen BASICODE-Programme über den Sprungverteiler des Z1013, welcher in der Zeitschrift "Funkamateur", Heft 10/88 veröffentlicht wurde. Der 64k-Bascoder trägt ab 0E000H das headersavekompatible Kassettenprogramm *maintape* ein, welches durch seine Befehle HLOAD und HSAVE genutzt wird. BASICODE-Programme werden mit dem Typkennzeichen "$" auf Kassette abgespeichert. Im Unterschied zum KC-BASIC wird ein bereits im Speicher vorhandenes Programm bei HLOAD nicht überschrieben. Dadurch lassen sich Programmteile einzeln entwickeln bzw. aus bereits vorhandenen Programmen herauslösen und anschließend zusammenfügen. Wird dieses nicht gewünscht, dann muß das bereits vorhandene Programm mit NEW gelöscht werden.

Bitte passen Sie das Programm *maintape* vor der Verwendung der entsprechenden Kommandos an Ihre Rechnerversion an. Die Anleitung dafür ist am Ende dieses Abschnittes zu finden.

Die DELETE-Taste besitzt je nach gültigen Mode unterschiedliche Bedeutung. Im Direktmode reagiert sie wie bisher, d. h., das durch den Cursor bedeckte Zeichen wird gelöscht. Im Programmode wird das Zeichen vor dem Cursor gelöscht. Damit werden sowohl die BASICODE-Anforderungen erfüllt als auch die bisherigen Editiermöglichkeiten beibehalten.

Mit LIST#2 lassen sich BASICODE-Programme über einen angeschlossenen Drucker ausgeben. Auch hier wird der Sprungverteiler vorausgesetzt.

Zusammenfassung wichtiger Fehlermeldungen

Die folgenden Fehlermeldungen erscheinen beim 64k-Bascoder oftmals mit der Fragestellung

```
WIEDERHOLEN J/N ?
```

Wird das Wiederholen abgewiesen, schließt sich bei Programmen eine Textkorrektur analog BASCON an, in deren Folge nochmals über

```
EINLESEN (E) ODER ABBRECHEN (A)?
```

entschieden werden kann.

kein Bestand

Es sollte ein Bestand über GOSUB 540 eingelesen werden. Auf der Kassette befand sich aber ein Programm.

kein Programm

Es sollte ein Programm mit dem 64k-Bascoder über "BCLOAD" eingelesen werden. Auf der Kassette befand sich aber ein Bestand.

Textpuffer leer

Es ist kein Programm zum Abspeichern vorhanden.

Blocknummer zu klein
Über GOSUB 540 wurde ein Bestand eingelesen, dessen Folgenummer kleiner als erwartet war. Betätigen Sie die ENTER-Taste und wiederholen das Einlesen, ohne die Kassette zurückzuspulen.

Blocknummer zu groß
Über GOSUB 540 wurde ein Bestand eingelesen, dessen Folgenummer größer als erwartet war. Spulen Sie die Kassette zurück und wiederholen das Einlesen.

Tastatur-Break
Das Einlesen oder Abspeichern im BASICODE-Format wurde durch Tastaturbetätigung unterbrochen.

Zeitfehler
Der Vorton eines BASICODE-Files war länger als erwartet.

Lesefehler
Das eingelesene Prüfbyte unterscheidet sich von dem Prüfbyte, welches beim Einlesen berechnet wurde. Im günstigsten Fall wurde nur das Prüfbyte falsch gelesen. Wenn Sie das Einlesen nicht wiederholen, dann bietet Ihnen die Textkorrektur mögliche fehlerhafte Zeilen zur Korrektur an.

I/O-ERROR
Die Fehlermeldung erscheint bei einer der folgenden Möglichkeiten. Es wurde versucht,

- mehr als einen Bestand zu eröffnen,
- einen Bestand abzuschließen, der nicht eröffnet ist,
- aus einem Bestand zu lesen, der zur Ausgabe eröffnet ist,
- in einen Bestand zu schreiben, der zur Eingabe eröffnet ist,
- einen Bestand mit NF = 4 bis NF = 7 zu eröffnen (Diskettenarbeit nicht möglich),
- einen Bestand zu eröffnen, ohne NF$ anzugeben.

OM-ERROR
Der zur Verfügung stehende freie Speicherbereich reichte nicht zum Einlesen bzw. Übersetzen des vorliegenden BASICODE- Programms aus. Eventuell haben Sie vergessen, ein bereits im Speicher stehendes BASIC-Programm mit NEW zu löschen. Beim 64k-Bascoder ist der Umweg über BASCON sinnvoll, da über LOAD# 1"name" eine direkte Übersetzung von der Kassette aus erfolgt. Das übersetzte Programm ist kürzer als das BASICODE-Textfile. Der umgedrehte Weg ist anzuraten, wenn diese Fehlermeldung beim Kommando BCSAVE im 64k-Bascoder auftritt.

Ebenfalls ist es möglich, den Programmspeicher des Bascoders nach einem Kaltstart auf Adresse 300H zu vergrößern. Dazu muß die Frage "Memory-Size:" mit der Eingabe einer dezimalen Zahl bis 52992 beantwortet werden. Der Bascoder bleibt trotzdem noch voll funktionsfähig.

SN-ERROR
Beim Übersetzen des eingelesenen BASICODE-Programms wurde ein Fehler festgestellt, der zum Abbruch des Übersetzungsvorganges führte. Geben Sie die letzte auf dem Bildschirm dargestellte Zeile über die Tastatur fehlerfrei ein und setzen Sie das Übersetzen beim 64k-Bascoder mit dem Kommando BCLOAD# fort. Ein nochmaliges Einlesen erübrigt sich dadurch in der Regel.

Beim 16k-Bascoder muß die fehlerhafte Zeile im BASCON korrigiert werden, ehe das Einlesen in den Bascoder wiederholt werden kann.

Einlesen und Anpassen anderer BASIC-Programme

In vielen Fällen werden Sie darauf zurückgreifen, bereits vorhandene BASIC-Programme an BASICODE-3 anzupassen. Gehen Sie dabei wie folgt vor:

Das BASIC-Programm wird mit Ihrem KC-BASIC über CLOAD oder HLOAD in den Z1013 eingeladen. Jetzt numerieren Sie alle Zeilennummern mit RENUMBER so um, daß es keine Zeilennummer mehr unter 1000 gibt. Sie können auch bereits alle DATA-Zeilen mit ihren späteren Zeilennummern versehen. Beim Einlesen in den Bascoder werden sie dann entsprechend umsortiert. Diese Vorgehensweise ist zwingend, wenn Sie nur über den 16k-Bascoder verfügen, da dessen Möglichkeiten eingeschränkt sind.

Das so vorbearbeitete Programm ist mit LIST#1"name" auf Kassette abzuspeichern. Es besitzt jetzt die gleiche Form wie nach der Konvertierung durch BASCON und kann in beide Z1013-Bascoder-Versionen mit LOAD#1"name" eingelesen werden.

Der 64k-Bascoder bietet die Möglichkeit, die Reihenfolge der Programmzeilen auch jetzt noch zu ändern. Geben Sie den entsprechenden Zeilen mit RENUMBER neue Nummern. Mit LIST ist zu sehen, daß sich die Zeilen trotz ihrer neuen Nummern noch an ihren alten Plätzen befinden. Jetzt übersetzen Sie das Programm mit BCSAVE in seine ASCII-Form. Die Aufforderung in der untersten Bildschirmposition

```
>>>>> AUFNAHME STARTEN <<<<<
```

ist mit ^C bzw. S4/K abzubrechen. Das alte Programm kann mit NEW gelöscht werden. Wird jetzt BCLOAD# eingegeben und die ENTER-Taste betätigt, dann wird das Programm aus dem Speicher, also ohne Umweg über die Kassette, neu übersetzt. Die vorher umnumerierten Zeilen erhalten so auch ihre neue Position. Zu beachten ist, daß dieser Ablauf nicht durch das EDIT-Kommando unterbrochen wird, da durch die zeitweise bestehenden, an sich absurden Zeilennummern Fehlabläufe entstehen können.

Der Bascoder prüft nach RUN folgende BASICODE-Bedingungen selbständig ab:

- Im Programm treten nur zulässige Befehle unter ihren besonderen Anwendungsvorschriften auf (z. B. Variablenzwang nach NEXT). Ausnahmen sind die genannten Editierkommandos der Z1013-Bascoder.
- Es wurden keine unzulässigen Variablen verwandt.

Wurden die Bedingungen nicht erfüllt, erscheint in der Regel die Fehlermeldung:

```
"SN-ERROR IN zlnr.".
```

Die entsprechende Zeile kann dann korrigiert werden.

Durch den Bascoder nicht geprüft wird der Umstand, ob vor und nach AND sowie OR Klammerausdrücke stehen, wie durch BASICODE verlangt wird. Diese Bedingung muß man selbst kontrollieren.

Besonderheiten des Z1013

Die Bascoder des Z1013 sind eigenständige, untereinander kompatible BASIC-Interpreter, in denen auch die Unterprogrammaufrufe GOTO 20, GOSUB 100 ... bis GOTO 950 integriert sind. Die entsprechenden Zeilen stehen physisch also gar nicht zur Verfügung. Sie sind nicht da. Die dazugehörigen Unterprogrammaufrufe werden als Befehle im internen Befehlsformat während des Einlesens, Eintippens bzw. Editierens generiert. Sie werden genau wie z. B. PRINT oder RUN realisiert. Das bedeutet aber auch, daß Programme, die mit RENUMBER auf Zeilennummern unter 1000 umnumeriert wurden, in der Regel noch voll lauffähig sind. Der RENUMBER-Befehl aktualisiert die Zeilennummern hinter GOTO und GOSUB direkt im Programm, also ohne nochmals eine

Einspeicherung vorzunehmen. Damit würde z. B. ein neu entstandenes GOSUB 100 als Befehl GOSUB mit der Zeilennummer 100 und nicht als Befehl GOSUB 100 im Programm stehen. Das gilt, solange diese Zeile nicht editiert wird.

Die Verwendung der genannten Unterprogrammaufrufe als Befehle hat den Vorteil, daß für jeden Aufruf dieser Zeilen nur ein Byte im Programmspeicher benötigt wird. Die Zeilen 20 bis 950 selbst nehmen keinen Platz im Programmspeicher weg. Das ist für die 16k-Version des Z1013 wichtig. Außerdem beschleunigt sich die Abarbeitungsgeschwindigkeit der BASICODE-Programme wesentlich.

Beide Bascoder des Z1013 sind nicht ROM-fähig. In den Routinen für Grafik und Musik werden sich selbst modifizierende Programmteile verwendet, die die Vorteile geringen Speicherplatzbedarfes und hoher Abarbeitungsgeschwindigkeit in sich vereinigen.

Einschränkungen bestehen aufgrund der mangelnden Grafikfähigkeit des Z1013. Da der Rechner im Normalfall keine inverse Schrift realisieren kann, setzt der Bascoder bei GOSUB 150 vor und hinter den Text einen Pfeil und gibt einen Aufmerksamkeitston über das Kassettenport aus.

Die grafische Schrift wird unabhängig von CN auf dem Bildschirm ausgegeben. Leerzeichen werden dabei ausgeblendet. Für CN gleich 1 prüft der Bascoder den vorhandenen Bildschirminhalt. Sind die auszugebenden Schriftzeichen bereits vorhanden, dann werden sie gelöscht, also mit Leerzeichen überschrieben.

Der Textschirm des Z1013 besteht aus 32 Zeilen mit je 32 Zeichen. Der Grafikschirm hat eine Auflösung von 48*64 Pixeln, nutzt also die oberen und unteren vier Zeilen nicht.

Die Tonausgabe (GOSUB 400 und Aufmerksamkeitston) erfolgt über das Kassettenport des Z1013 in einheitlicher Lautstärke. Bei SV = 0 hören Sie keinen Ton. Der Tonumfang reicht von SP = 0 bis SP = 119, das sind 10 Oktaven. Alle darüberliegenden Töne werden als SP = 119 ausgegeben (ca. 16kHz). Die Tondauer wird auf mehr als fünf Millisekunden genau bestimmt. Innerhalb der unteren Oktaven können bei kurzen Tönen Abweichungen auftreten, die auf Rundungsfehler zurückzuführen sind.

Die abbrechbare Warteroutine GOSUB 450 wurde im Zeitverhalten auf die Originaltastatur abgestimmt. Bei Verwendung einer anderen Tastaturabfrage können abweichende Werte für SP auftreten.

Die vierte Ebene der Originalfolienflachtastatur erhielt folgende Bedeutung:

Shift-4	ASCII	Bezeichnung	Control	Bedeutung
S4/@	10H	DEL	^P	Zeichen löschen (interne Codierung 7FH)
S4/A	11H	UA		Alpha-Umschaltung
S4/B	12H	INS	^R	Fügt Leerzeichen an Cursorposition ein
S4/C	13H	ESC	^S	Druckersteuerung ESCAPE
S4/D	14H	LIST	^T	Listet BASICODE-3-Programm
S4/E	15H	RUN	^U	Startet BASICODE-3-Programm
S4/G	17H	UG		Grafik-Umschaltung
S4/J	2	CL LN	^B	Löscht Eingabezeile
S4/K	3	BREAK	^C	Programmabbruch
S4/N	6	CONT	^F	Setzt unterbrochenes Programm fort
S4/P	8	←	^H	Cursor links (interne Codierung 1CH)
S4/Q	9	→	^I	Cursor rechts (interne Codierung 1DH)
S4/R	0AH	↑	^J	Cursor hoch (interne Codierung 1EH)
S4/S	0BH	↓	^K	Cursor runter (interne Codierung 1FH)
S4/T	0CH	CLS	^L	Bildschirm dunkel löschen
S4/U	0DH	ENT	^M	ENTER
S4/V	0EH/19H	I←	^N	Cursor an linken Rand
S4/W	0FH/18H	→I	^O	Cursor an rechten Rand

Aus der Tabelle geht ebenfalls die Belegung von Funktionstasten bei anderen Tastaturen hervor.

Anpassungen der Bascoder an unterschiedliche Ausrüstungsvarianten des Z1013

Die Tastatur wurde für die Verwendung eines A.2-Monitors gesondert entprellt. Wollen Sie bei Verwendung eines 2.02-kompatiblen Monitors die Repeatfunktion oder eine Stringtabelle nutzen, dann ersetzen sie bitte auf

```
(Ø132H) 28H in 18H.
```

Weitere Anpassungen (z. B. an genannten Sprungverteiler des Z1013) sind wie folgt möglich:

- Ansprung eines Druckertreibers auf (2AFAH),
- Ansprung eines gesonderten Aufmerksamkeitstones auf (2AFDH),
- Blinkfrequenz des Cursors auf (013FH)
  und auf (0142H),
- abgefragte Tastaturspalte bei BCLOAD und BCSAVE (64k-Bascoder)
  auf (319FH),
- abgefragte Tastaturspalte bei Einlesen und Abspeichern (BC)
  (BASCON) auf (0EFFH).

Das Programm *maintape* ist vor Verwendung der Kommandos HLOAD und HSAVE im 64k-Bascoder an Ihren Rechner anzupassen. Für den A.2-Monitor sind folgende Änderungen vorzunehmen:

- auf (32AFH), (32C6H) und (3396H) "2AH 49H 0" in "21H 0 0ECH"
- auf (36E6H) "0C3H" in "0C9H"
- auf (336EH) "37H" in "0B7H"

Für den 2.02-Monitor ist zu korrigieren:

- auf (36E6H) "0C3H" in "0C9H"
- auf (336Eh) "37H" in "0B7H"

Für den 4k-Monitor mit Sprungverteiler entsprechend der Veröffentlichung in der Zeitschrift "Funkamateur", Heft 10/88 ist zu modifizieren:

- auf (336EH) "37H" in "0B7H"

Daten des Z1013 im BASICODE-Zustand

Taktfrequenz	(MHz)	2
BASIC-ROM	(kByte)	-
Betriebssystem	(kByte)	2 bzw. 4
Freier RAM min.	(kByte)	5
Freier RAM max.	(kByte)	32
Bildschirmzeilen		32
Zeichen je Zeile		32
Grafikzeilen		48
Grafikpunkte je Zeile		64
SP tiefster Ton		0
SP höchster Ton		119
Rechengenauigkeit		7
Kleine Buchstaben		ja

5.7. Bascoder für den Amateurcomputer AC 1

Vorbemerkungen

Im Gegensatz zu allen anderen hier behandelten Computern ist der AC 1 [20] ein reines Eigenbaugerät und somit nicht handelsüblich. Das trifft auch größtenteils auf die dazu verwendete Software zu. So auch auf die Vielzahl der BASIC-Interpreter, die auf dem AC 1 verwendet werden. Damit hätte es wenig Sinn, einen Bascoder zu einem dieser Interpreter zu schreiben, zu den anderen würde er kaum passen, Mißverständnisse wären die Folge.

Wir nahmen deshalb einen 8K-BASIC-Interpreter-Kern, der in seiner Syntax den Vorschriften von BASICODE bereits weitestgehend entsprach, erweiterten ihn um einige Befehle und fügten die für BASICODE notwendigen Befehle und Routinen an [16...19]. Der daraus entstandene Interpreter gestattet es nun einerseits, BASICODE-Programme auszuführen und zu erstellen, und ist andererseits zu den auf unseren Klein- und Heimcomputern verwendeten Interpretern größtenteils kompatibel. Beides gilt mit kleinen Einschränkungen. Sie betreffen einerseits die begrenzten Graphikmöglichkeiten des AC 1 und bezüglich der hier vorgestellten Version, die noch nicht enthaltene Dateiarbeit. BACOBAS ist unabhängig von der installierten Monitorvariante, sofern diese die Restartroutinen und den Sprungverteiler entsprechend [20] enthält.

Hardwarevoraussetzungen

Bevor nun genauer auf die Software eingegangen wird, ist es notwendig, einige Worte zur erforderlichen Mindestausstattung des AC 1 zu verlieren. Notwendig sind:

- 2 kByte Bildwiederholspeicher
- Zeichengenerator mit den Elementen für die Graphikbefehle (z. B. [20])

- Tastaturanschluß entsprechend [20]
- Tonausgabe auf PB6 (Kassettenport)
- RAM-Erweiterung ab 2000H, möglichst bis 64K.

Sind diese Punkte nicht vollständig erfüllt, so läuft BACOBAS entweder gar nicht oder mit Einschränkungen.

BASICODE mit BACOBAS

BACOBAS benötigt ab 3000H RAM-Speicher und ist je nach Version maximal 12 kByte lang. Der Kaltstart erfolgt auf Adresse 3000H und der Warmstart auf der Adresse 3003H. Wer sich das Programm erst einmal kopieren möchte, sollte dazu das Kopierprogramm "Copy" benutzen, da sonst das Titelbild und die Autostartfunktion verlorengehen.
Der Interpreter wird wie jedes andere Maschinenprogramm mit L <CR> eingelesen. Zuerst erscheint das Titelbild, welches in Abbildung 5.14 dargestellt ist. An dieser Stelle möchten wir Herrn Jürgen Hannemann für seine Unterstützung bei der Gestaltung dieses Bildes danken.

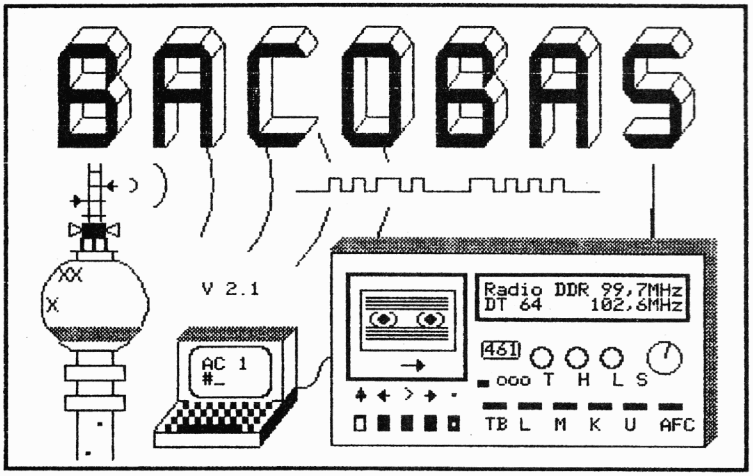


Abbildung 5.18. Startbild AC1

Nach einer Minute und etwa 20 Sekunden ist das Einlesen beendet und das Programm startet automatisch. Der Bildschirm wird gelöscht, es werden die Überschrift und die Frage nach der Speichergröße ausgegeben, die man im allgemeinen mit <CR> beantwortet. Danach befindet sich BACOBAS im normalen BASIC-Mode, der immer durch einen vollausgemalten ganzen oder halben Cursor angezeigt wird.

```
BACOBAS V. 2.1

Standard BASIC by Microsoft (c) 1978
Ext. & Bascoder by F&B soft (c) 1989

Memory end?
xxxxx Bytes free

ok
>  (ganzer weißer Cursor)
```

Der Cursor hat in BACOBAS drei Anzeigefunktionen, die dem Anwender Auskunft über den aktuellen Zustand geben sollen. Erstens zeigt er an, ob man sich im normalen BASIC-Mode (BASIC-Schlüsselworte existieren als Token) oder im BASICODE-Mode (Subroutinenblock im Speicher) befindet. Zweitens gibt der Cursor Auskunft darüber, ob sich die Tastatur im Computer- (Großbuchstaben ohne Shift) oder Schreibmaschinenmodus (Kleinbuchstaben ohne Shift) befindet und drittens ob sich ein ASCII-File im BASICODE-Puffer befindet. Daraus ergeben sich folgende fünf Möglichkeiten:

- BASIC, Computermode	Cursor weiß, ganz
- BASIC, Schreibmaschinenmode	Cursor weiß, halb
- BASICODE, Computermode	Cursor grau, ganz
- BASICODE, Schreibm.-mode	Cursor grau, halb
- ASCII-File im Puffer	Cursor weiß mit Punkt

Ob nun ein gewöhnliches BASIC-Programm oder ein BASICODE-Programm bearbeitet wird, hängt ausschließlich von den nachfolgenden Bedienhandlungen ab. Für die BASICODE-Bearbeitung enthält BACOBAS folgende Steuerwörter:

BACO	BASICODE-Mode einstellen
CONV	BASIC-Programm in BASICODE wandeln
BSAVE	gewandeltes Programm in BASICODE-Codierung auf Kassette ausgeben
BLOAD	BASICODE -Programm von Kassette einlesen
TRANS	BASICODE-Programm in BASIC-Tokens wandeln
BLIST	BASICODE-Puffer auf Bildschirm listen
BEDIT	BASICODE-Puffer editieren

Im folgenden sollen diese Kommandos genauer beschrieben werden.

BACO

Damit wird der Interpreter in den BASICODE-Mode gesetzt. Diese Funktion wird immer dann genutzt, wenn man selbst ein Programm entsprechend den BASICODE-Regeln erstellen will. Der Cursor wechselt von weiß nach grau. Ein im Speicher vorhandenes Programm wird gelöscht, der Subroutinenblock wird in den Speicher gebracht. Die Kommandos LIST, AUTO und EDIT funktionieren erst ab Zeile 1000, RENUMBER erst ab Zeile 1010. CSAVE speichert alles ab Zeilennummer 1000 auf Kassette, um nicht Zeit und Band mit dem Subroutinenblock zu verschwenden. Ein auf erste Zeilennummer = 1010 bereits vorher umnumeriertes und abgespeichertes normales BASIC-Programm kann nun z. B. mit CLOAD eingelesen und dann zur Weitergabe in BASICODE bearbeitet werden.

Dieser Mode kann entweder durch NEW oder durch BYE und anschließenden Kaltstart verlassen werden.

CONV

CONV wandelt ein BASIC-Programm, das Token enthält, ab Zeilennummer 1000 in ASCII um, damit dieses dann in BASICODE-Codierung abgespeichert werden kann. Diese Umwandlung geschieht in zwei Läufen. Im ersten Lauf wird geprüft, ob nur im BASICODE zugelassene Schlüsselworte enthalten sind und ob die zulässige Zeilenlänge eingehalten wird. Nur wenn alles fehlerfrei ist, erfolgt im zweiten Lauf eine Umwandlung. Jede bearbeitete Programmzeile wird in der untersten Bildschirmzeile kurz ausgegeben, fehlerhafte Zeilen werden mit der entsprechenden Fehlermeldung nach oben gerollt. Falls mehr Fehler auftreten sollten, als auf dem Bildschirm Platz finden, so kann CONV mit Ctrl-C abgebrochen werden. Die Fehler sind mit Hilfe des EDIT-Kommandos zu beseitigen. CONV geht nur im BASICODE-Mode (grauer Cursor) und auch nur dann, wenn die

Zeile 1000 vorhanden ist. Da bei der Umwandlung der Token in ihre ausgeschriebene ASCII-Darstellung das Programm länger wird, kann es bei langen Programmen passieren, daß der Speicher nicht mehr ausreicht. Dann erfolgt die Fehlermeldung "OM Error" und die Umwandlung unterbleibt. Hier eine mögliche Fehlermeldung:

```
>CONV
167Ø SET (25,1Ø)
^ unerlaubter Befehl!
258Ø PRINT"Diese Zeile hat zu viele Sterne
************************"
^ Zeile zu lang!
1. Lauf: Ø2 Fehler
ok
>
```

Die erfolgreiche Ausführung des CONV-Kommandos wird vom Interpreter mit der Ausschrift "ASCII-File im Puffer" bestätigt. Außerdem wird nun ein weißer Cursor mit Punkt angezeigt. Immer, wenn dieser Cursor erscheint, werden alle Befehle, die den BASIC-Speicher beeinflussen, verweigert.

Sinnvolle Befehle wären jetzt BLIST, BSAVE oder TRANS.

BSAVE

Mit BSAVE wird der Inhalt des Pufferspeichers in der BASICODE-Codierung auf Kassette abgespeichert, d. h., bevor BSAVE mit < CR > abgeschlossen wird, ist der Recorder zu starten. Ist kein fehlerfreies ASCII-File (weißer Cursor mit Punkt) im Puffer, wird BSAVE verweigert.

BLOAD

BLOAD dient zum Einlesen von Programmen in BASICODE-Codierung. Für das Einlesen ist es ohne Bedeutung, ob BLOAD durch das Drücken von < CR > vor oder während des Kenntones gestartet wird. Das Einlesen kann in der untersten Bildschirmzeile verfolgt werden, wobei allerdings ein Mitlesen infolge der hohen Übertragungsgeschwindigkeit kaum möglich ist, das Hochzählen der Zeilennummer ist aber gut zu beobachten.

Erfolgt nach dem Einlesen keine Fehlerausschrift, dann stimmt die berechnete Prüfsumme mit der eingelesenen überein. Mit gewisser Sicherheit liegt dann kein Fehler vor. Der Cursor ist jetzt wieder weiß mit Punkt.

Auch hier wird alles unterbunden, was den Pufferspeicher verändern könnte. Nach BLOAD sind die Kommandos BLIST, BSAVE, TRANS und BEDIT möglich. Das Kommando BEDIT sollte jedoch wirklich nur als "Notbremse" bei Einlesefehlern gebraucht werden.

TRANS

Mit diesem Kommando wird der Inhalt des BASICODE-Puffers in die Form umgewandelt, die der BASIC-Interpreter abarbeiten kann. Das heißt, alle Schlüsselwörter werden durch Token ersetzt und der Subroutinenblock wird vorangesetzt. Dieses Kommando ist die Umkehrung von CONV. TRANS wird nur dann ausgeführt, wenn der BASICODE-Puffer geladen ist, also entweder nach BLOAD oder nach CONV. Man erkennt das auch am weißen Cursor mit dem schwarzen Punkt. Auch bei TRANS wird jeweils die gerade bearbeitete Zeile auf dem Bildschirm kurz ausgegeben. Damit kann man sich von der ordnungsgemäßen Arbeit des Interpreters überzeugen und kommt bei längeren Programmen nicht auf den Gedanken, der Rechner sei abgestürzt. Nach der Ausführung von TRANS erscheint dann der graue Cursor. Nun kann das Programm z. B. durch RUN gestartet werden.

BLIST

Mit BLIST ist es möglich, den Inhalt des BASICODE-Puffers auf dem Bildschirm auszugeben. Es funktioniert genauso wie der LIST-Befehl im Interpreter und kann mit Ctrl-C abgebrochen werden.

BLIST <CR> listet ab Zeile 1000
BLIST "Zeilennummer" listet ab Zeilennummer, die durch die Zeichenkette "Zeilennummer" gekennzeichnet ist

Nach der Ausgabe der mit LINES vereinbarten Zeilenanzahl stoppt die Anzeige. Mit <CR> wird fortgesetzt.

BEDIT

Dieses Kommando ermöglicht es, im BASICODE-Puffer zu ändern, man sollte davon jedoch nur bei Lesefehlern Gebrauch machen, die die Zeilennummer beschreibt bzw. die den Zeilenanfang kennzeichnet. Diese Zeichenkette darf maximal eine Bildschirmzeile lang sein. Ist in der Zeichenkette ein " enthalten, so wird danach nicht mehr zwischen Groß- und Kleinbuchstaben unterschieden. BEDIT gestattet auch, Zeilen bzw. Zeilennummern zu streichen, dies ist jedoch wirklich nur zu verwenden, um nach Lesefehlern einen ordnungsgemäßen Ablauf von TRANS zu ermöglichen. Ansonsten sollte man nach TRANS mit Hilfe von EDIT arbeiten. BEDIT muß eine Zeichenkette folgen, die die Zeilennummer beschreibt. Auch hierfür zwei Beispiele:

BEDIT "1050 <CR> Editieren der Zeile 1050
BEDIT "2XD0 <CR> Editieren der Zeile 2XD0
↑↑
Lesefehler

In BEDIT existieren die folgenden Steuerzeichen:

Ctrl-H	Cursor links	Ctrl-I	Cursor rechts
Ctrl-E	Zeichen einsetzen	Ctrl-S	Zeichen streichen
Ctrl-R	rückwärts löschen		

Das Editieren einer Zeile wird mit <CR> abgeschlossen oder mit Ctrl-C abgebrochen.

Ausgewählte Arbeitsabläufe

Um die Arbeit mit diesen speziellen Kommandos für BASICODE zu verdeutlichen, sind nachfolgende Standardabläufe kurz aufgelistet.

1. BASICODE-Programm einlesen, abarbeiten, bearbeiten, testen und speichern:

```
>BLOAD              ; Einlesen in BASICODE-Codierung
>BLIST              ; Auflisten des BASICODE-Puffers
:
:
>BEDIT"2X8Ø"        ; Beseitigen eines Lesefehlers
:
>TRANS              ; Transformieren in die Token-Form
>RUN                ; Starten eines Programms
:
:
>LIST               ; Auflisten des BASIC-Puffers
:
:
```

```
>EDIT 545Ø          ; Ändern einer Zeile
:
>CONV               ; Convertieren in den BASICODE-Puffer
1. Lauf: ØØ Fehler
ASCII-File im Puffer
ok
>BSAVE              ; Ausgeben in BASICODE-Codierung
```

2. Copieren eines BASICODE-Programms

```
>BLOAD
>BSAVE
```

3. Erstellen eines BASICODE-Programms

```
>BACO
>AUTO
:
Programm eingeben
:
>RUN
:
>EDIT 225ØØ
:
>LIST
:
>CONV
1. Lauf: ØØ Fehler
ASCII-File im Puffer
ok
>BSAVE
```

4. Abspeichern bzw. Einlesen eines BASICODE-Programms im AC 1-Kassettenverfahren zur Zeiteinsparung:

a) Der Interpreter befindet sich im BASICODE-Mode, also befinden sich sowohl der Subroutinenblock als auch das eigentliche Programm im Speicher.

```
>CSAVE"TEST"
```

Damit wird das Programm ab Zeile 1000 unter dem Namen "TEST" abgespeichert.

b) Auf Band befindet sich ein nach a) gespeichertes BASICODE-Programm oder ein auf 1. Zeile = 1000 "renumbertes" normales BASIC-Programm.

```
>BACO
>CLOAD
```

Fehlermeldungen der BASICODE-Kommandos

?TR-Error	Kein gültiges ASCII-File im Puffer erkannt.
?BC-Error	Im BASICODE-Mode wurde versucht, auf eine Zeile < 1000 zuzugreifen.
?CO-Error	Im BASIC-Mode wurde versucht zu convertieren oder Zeile 1000 fehlt.
Lesefehler	Das berechnete Prüfbyte stimmt nicht mit dem Eingelesenen überein.

^ unerlaubter Befehl! In der Zeile ist ein Befehl enthalten, der nicht in BASICODE erlaubt ist.
^ Zeile zu lang! Die Zeile ist länger als 60 Zeichen.

Ist BASICODE nicht fehlerfrei oder endete BLOAD mit "Lesefehler", so wird BSAVE verweigert

Kurze Zusammenstellung der Leistungsfähigkeit von BACOBAS

Alle nachfolgend nicht näher erläuterten Kommandos und Funktionen entsprechen in ihrer Syntax dem auf unseren Kleincomputern installierten BASIC-Interpreter (KC85, KC87). Die bereits erläuterten speziellen Kommandos für BASICODE sind nicht noch einmal aufgeführt.
Kommandos:

AUTO, BYE, CLEAR, CLOAD, CLS, CONT, CSAVE, EDIT, LINES, LIST, RENUM, RUN, TROFF, TRON, WIDTH

Befehle:

BEEP, CALL, DATA, DEEK, DEFFN, DIM, DOKE, END, FOR...TO...STEP, GOSUB, GOTO, IF..THEN, INP, INPUT, LET, NEW, NEXT, ON...GOTO, ON...GOSUB, OUT, PEEK, POKE, PRINT, READ, REM, RESTORE, RETURN, STOP, WAIT

Als Sonderbefehle existieren:

BROFF	Abbruch mit Ctrl-C ausgeschaltet
BRON	Abbruch mit Ctrl-C eingeschaltet
LLIST	Listen des Programms auf einem Drucker
LPRINT	Ausgabe wie PRINT, aber auch auf den Drucker

Und die Grafikbefehle:

POINT, RESET, SCREEN, SET. Sie werden unten erklärt.

Operatoren:

(), ^ (Exponent), *, /, +, -, =, <, >, < >, < =, > =, NOT, AND, OR

Funktionen:

ABS, ATN, COS, EXP, INT, LN, SGN, SIN, SQR, TAN

Stringbehandlung:

LEFT$, LEN, MID$, RIGHT$, STR$, VAL

Sonstige Funktionen:

ASC, CHR$, FRE, POS, RND, SPC, TAB, NUL

Die Fehlermeldungen entsprechen denen der üblichen BASIC-Interpreter [xx].

Hinweise zur Bedienung und zu einigen speziellen Befehlen

Start, Restart und Unterbrechung

Start:	J 3000 <CR>
Restart:	J 3003 <CR>; nur wenn vorher Start erfolgte! (z. B. nach "BYE")
Unterbrechung:	CTRL-C (z. B. nach RUN oder zum Beenden von AUTO)

Cursorpositionierung

Mit Hilfe des Befehles SCREEN (x,y) ist es möglich, den Cursor auf eine der 2048 möglichen Plätze des Bildschirmes zu setzen. Damit läßt sich dann z. B. die Rollfunktion des Bildschirmes unterdrücken. "x" enthält die Nummer der Spalte (0...63) und "y" die Nummer der Zeile (0...31), wobei der Ursprung in der linken oberen Ecke des Bildes liegt.
Beispiel:

```
1Ø SCREEN (3Ø,16): PRINT"Test" : REM druckt ca. in Bildmitte
2Ø SCREEN (3Ø,16): PRINT" " :    REM loescht wieder
```

Pseudographik

Mit Hilfe von Graphikelementen im Zeichengenerator (Codes 00H - 0FH) ist es möglich, Felder (3*4 Punkte) auf dem Bildschirm lückenlos im Raster 128*64 zu setzen, zu testen und zurückzusetzen. Damit lassen sich Kurvenverläufe schon relativ gut darstellen.

Der Ursprung des Koordinatensystems liegt in der linken oberen Ecke des Bildschirmes.

```
0.............123 ←x
:               :
:               :
:               :
63..............
↑
Y
```

SET (x,y)	setzt das Feld x,y auf weiß
RESET (x,y)	setzt das Feld x,y zurück, d.h. auf schwarz
POINT (x,y)	liefert die Information "wahr" für Feld gesetzt und "falsch" für ein rückgesetztes Feld

Die Argumente von SCREEN, SET, RESET und POINT dürfen nur Integerwerte annehmen. Ist das nicht der Fall oder werden die zulässigen Bereiche überschritten, erfolgt eine Fehlermeldung.

Druckerbefehle

Zur Ausgabe auf einen vorhandenen Drucker wurde die Befehle LLIST und LPRINT aufgenommen. Da AC 1 als reiner Eigenbaucomputer natürlich nicht über ein spezielles Druckerinterface verfügt, sondern jeder den Drucker nimmt, der für ihn greifbar ist, ist es notwendig, daß auch jeder selbst die am Schluß beschriebenen Druckerroutinen installiert. Ansonsten ist dafür der Aufruf der Zeitschleife MS30 als Dummy eingesetzt.
LLIST setzt jedem Listing des Basic-Programms die Überschrift

```
AC 1 BASIC - LISTING
```

voran.

LPRINT gibt parallel auf dem Schirm und dem Drucker aus. Hier sei angemerkt, daß eine Eingabe in der Form L? natürlich nicht LPRINT abkürzt. Das führt zu einer Fehlinterpretation.

Abspeichern auf Kassette

Abspeichern: CSAVE"Name" Name = Programmname
Einlesen: CLOAD

Einlesen lädt das nächste Programm von Kassette hinter das bereits im Speicher vorhandene, d.h., soll ein neues Programm eingelesen werden, ist das alte durch NEW zu löschen. Wie auch im Monitor, sollte dann die CR-Taste erst nach Beginn des Kenntones gedrückt werden. Nach dem Ende der Kassettenaufzeichnung muß auf dem Bildschirm sofort das BASIC-Prompt ">" erscheinen. Ansonsten hat sich ein Lesefehler eingeschlichen. Dann hilft meist nur NEW und ein erneuter Leseversuch. Wenn das Kassetteninterface im Monitor einwandfrei arbeitet, und das muß der Normalfall sein, gibt es auch im BASIC keine Probleme mit CSAVE und CLOAD.

Editieren einer Zeile

Mit dem EDIT-Befehl ist es möglich, Zeileninhalte und Zeilennummern zu ändern, zu streichen, einzufügen bzw. zu doppeln.

Der Aufruf erfolgt mit EDIT Zeilennummer. EDIT wird mit <CR> beendet oder mit Ctrl-C abgebrochen.

Es werden folgende Steuerzeichen verwendet:

Ctrl-H	Cursor links	Ctrl-I	Cursor rechts
Ctrl-T	Cursor hinter Zeilennummer	Ctrl-R	rückwärts löschen
Ctrl-E	Zeichen einfügen	Ctrl-S	Zeichen streichen
Ctrl-W	Cursor zum Anfang des nächsten Befehls	Ctrl-F	Shift-lock ein/aus

Tonausgabe

Mit BEEP x, y, z ist es möglich, einen Ton einer bestimmten Frequenz, Dauer und Lautstärke zu erzeugen. Die Tonausgabe erfolgt am Kassettenport PB6. Zur Steuerung der Lautstärke wird auf den untersten 4 Bit einer zu vereinbarenden E/A Adresse ein dementsprechendes Bitmuster ausgegeben, wobei das Muster 1110 dem leisesten Ton und 0000 dem lautesten Ton entspricht. Die entsprechende E/A-Adresse ist auf Adresse 3006H einzutragen, und ein Netzwerk zur Lautstärkesteuerung muß dann in die Tonausgabeleitung eingefügt werden. In Abb. 2 ist ein Schaltungsvorschlag zur Lautstärkestellung angegeben. Der Wert des Grundwiderstandes R ist dabei der Impedanz des nachgeschalteten Verstärkers anzupassen.

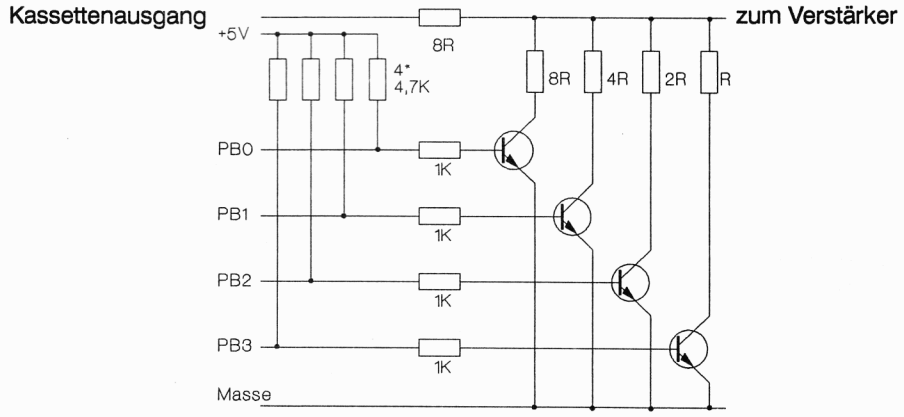


Abbildung 5.19. Netzwerk zur Pegeleinstellung

Man kann aber auch einfach auf die Lautstärkestellung verzichten. Die Parameter für Frequenz, Dauer und Lautstärke entsprechen den in BASICODE vereinbarten.
Für die Tonhöhe sind die Argumente von 0 bis 127 zulässig, aber nur die Werte von ca. 20 bis 101 erscheinen sinnvoll. Unterhalb von 20 läßt die Tonqualität doch zu wünschen übrig, da ja nur Rechteck- und keine Sinusschwingungen erzeugt werden. Oberhalb 101 besteht in zunehmendem Maße nicht mehr für alle Argumente ein akustischer Unterschied in der Tonhöhe, da hier der gewählten Form der Tonerzeugungssoftware Grenzen durch die Taktfrequenz gesetzt sind. Diese Extremwerte dürften aber auch für Musikausgaben keine Bedeutung haben.
Beispiel:

```
5Ø BEEP 69,1Ø,7
6Ø REM erzeugt 1 s den Kammerton a (44Ø Hz) in mittlerer Lautstärke
```

Einige wichtige Adressen von BACOBAS

3000H	Kaltstart
3003H	Warmstart
3006H	E/A-Adresse zur Lautstärkeeinstellung
3007H	Sprung zur eigenen Druckerausgaberoutine; die Zeichenübergabe erfolgt im Akku, weitere Register dürfen nicht zerstört werden .
300AH	Sprung zur Initialisierung des eigenen Druckertreibers; es darf kein Register zerstört werden. Dieses Programm wird bei jedem LLIST und nach RUN vor den ersten LPRINT aufgerufen. Ist kein eigener Druckertreiber vorhanden, so brauchen diese beiden Sprungbefehle nicht verändert zu werden, sie rufen dann nur die Routine MS 30 des Monitorprogramms auf.

5.8. Bascoder für ATARI 800XL/XE 600XL 130XE

Allgemeines

Da bislang für die ATARI-Rechner kein Bascoder zur Verfügung stand, waren bei der Entwicklung zwei Aspekte wesentlich:

Erstens sollte das Programm schnell entwickelt werden, um möglichst bald den ATARI-Nutzern die Mitarbeit bei BASICODE zu ermöglichen und den Software- und Datenbestand von BASICODE zugänglich zu machen.

Zweitens sollte die verwendete Hardware so einfach wie möglich und die Änderungen an Geräten so gering wie möglich sein.

Unter diesen Prioritäten hat an einigen Stellen die "Schönheit" des Programms etwas gelitten. An der Zuverlässigkeit und Portabilität wurden jedoch keine Abstriche gemacht.
Auf zwei Variablen der Hardware-Änderungen wird am Ende eingegangen.

Für ganz Ungeduldige

Wenn die Datasette wie hinten beschrieben umgebaut ist, kann man bereits mit BASICODE arbeiten. Dazu wird

die Kassette mit dem Bascoder eingelegt,
der Recorder auf "A" geschaltet,
der Recorder gestartet,
die START- und die OPTION-Taste gedrückt und
der Rechner eingeschaltet.

Nachdem sich BASICODE gemeldet hat, wird der Recorder auf "B" geschaltet, der Befehl

```
LOAD "B"
```

eingegeben und bei Ertönen des BASICODE-Vorlauftons eine Taste gedrückt. Der Bildschirm muß viermal dunkel werden, nach der darauffolgenden READY-Meldung ist BASICODE betriebsbereit.

Jetzt kann von der Kassette ein BASICODE-Programm mit

```
CL.
```

eingelesen und danach mit RUN gestartet werden.

Im Rechner stehende BASICODE-Programme lassen sich mit

```
CS.
```

auf Kassette schreiben.

Software

Für den ATARI-Rechner stehen vor allem das ATARI-BASIC und das TURBO-BASIC XL zur Verfügung. Beide BASIC-Dialekte weichen in mehreren Punkten stark von den üblichen MICROSOFT-BASIC und allen daraus abgeleiteten Varianten ab. Das Grundprinzip dieser Interpreter liegt jedoch der Struktur von BASICODE zugrunde. Deshalb kamen die o. g. ATARI-BASIC nicht für BASICODE in Betracht. Nun gibt es zwar für den ATARI auch ein MICROSOFT-BASIC, aber dieses ist so groß, so schwerfällig und teilweise fehlerbehaftet, daß sich das BASICODE-Testprogramm (17K) nicht mal mehr laden, geschweige denn abarbeiten ließ. Deshalb wurde für den ATARI ein bei anderen Rechnern unüblicher Weg beschritten.

Es wurde insgesamt für BASICODE ein spezieller BASIC-Interpreter geschrieben, der zugleich die Arbeitsweise von BASICODE (Lesen und Schreiben von BASICODE-Programmen, Bearbeiten von BASICODE-Datenbeständen usw.) als seinen normalen Standard ansieht.

Dieser Interpreter bearbeitet alle in BASICODE zugelassenen BASIC-Befehle und -Funktionen. Er kann aber auch als neuer BASIC-Interpreter selbständig benutzt werden. Dabei gibt es die folgenden Besonderheiten:

AND; NOT; OR

Die Operationen werden bitweise ausgeführt.

INPUT

Der Befehl darf nur im Programm benutzt werden, im Direktmodus ist die Zuweisung zu benutzen. INPUT läßt (erweitert gegenüber BASICODE-3) mehrere Variablen, die mit Komma getrennt sind, zu.

RESTORE

Ebenfalls über BASICODE-3 hinaus, ist hinter RESTORE auch eine Zeilennummer zulässig. Weiterhin seien alle Funktionen und Kommandos vom neuen ATARI-Interpreter, die zusätzlich zu dem in diesem Buch beschriebenen BASICODE-3-Befehlsvorrat aufgenommen wurden oder abweichendes Verhalten zeigen, alphabetisch aufgeführt:

C. **C.(I)**
Funktion zum Schließen einer Datei. Sie nimmt einen logischen Wert an, =0 falls o.k., < >0 im Fehlerfall. I ist die logische Kanalnummer des Files (s. O.-Befehl).

CL. **CL.**
Laden von BASICODE. Hier wird bei den folgenden Kommandos der Punkt Pflicht. Dieses Kommando führt folgende Schritte durch:

1. Das Programm ab Zeile 1000 wird gelöscht.
2. Die Datasette wird eingeschaltet. Bei Ertönen des BASICODE-Vorlauftons ist eine Taste zu drücken, danach wird das Programm eingelesen. Während dieser Zeit ist der Bildschirm abgeschaltet. Soll der Lesevorgang abgebrochen werden, ist START zu drücken.
3. Nach Beendigung des Lesens wird der Recorder vom Rechner angehalten, das Bild erscheint wieder. Jetzt setzt der Interpreter das eingelesene Programm in seine interne Darstellung um, dies kann bei langen Programmen schon einige -zig Sekunden dauern. Mit READY wird das Ende der Aktion angezeigt. Ein eventuell aufgetretener Prüfsummenfehler wird erst jetzt gemeldet. Dies gilt für den Normalfall, also sofern der Lesefehler nicht so folgenschwer war, daß die Grundstruktur des Programms zerstört wurde.

CLR **CLR**
Alle numerischen Variablen werden auf Null und die Zeichenketten auf leer gestellt. Reservierter Speicher wird freigegeben.

CONT **CONT**
Die Programmdurchführung wird fortgesetzt, sofern die Unterbrechung zuvor durch die BREAK-Taste oder eine STOP- bzw. END-Anweisung ausgelöst wurde. Sie wird an dem Punkt wieder aufgenommen, an dem die Unterbrechung erfolgt ist. Nach einem Fehler mit Halt kann ebenfalls die Bearbeitung fortgesetzt werden.

CS. **CS.**
Das vorhandene Programm wird ab Zeile 1000 im BASICODE-Format auf Kassette gespeichert.

D. **D. I, J, K**
Zeichnet im grafischen Betrieb eine Linie der Farbe K vom Punkt des letzten P.-Befehls oder dem Endpunkt des letzten D.-Befehls zum angegebenen Punkt I, J. Zur Bedeutung von I, J und K siehe den Befehl P.

END **END**
Die Programmdurchführung wird beendet, alle Dateien werden geschlossen und zum Befehlsniveau zurückgekehrt.

F **F. (X)**
Bringt die Anzahl von Bytes zurück, die nicht vom Programm genutzt werden. F. (X) verdichtet den Speicher mit der garbage collection, dies kann aber bis zu einer Minute dauern!

G **G.**
Schaltet Grafikbetrieb ein. In ATARI-BASICODE wird nur ein Grafikmodus verwendet. Er entspricht Modus 24 in ATARI-BASIC. Es werden 192 Zeilen je 320 Bildpunkte dargestellt. Der Hintergrund ist weiß, der Vordergrund schwarz.

INPUT **INPUT [<"prompt">;]<variablenliste>**

Eingabe vom Terminal. Zur Kennzeichnung der Eingabe kann <"prompt"> angegeben werden, diese Kette wird als Bedienernachricht ausgegeben. Danach wird ein Fragezeichen ausgegeben. Die Eingabe wird erst beendet, wenn die <variablenliste> vollständig gefüllt ist. Ein vorheriger Abbruch oder Fehler (Typ, Länge usw.) wiederholen die Eingabeanforderung.

LIST **LIST [<zeile>[-]]**
LIST [<zeile1>[- [<zeile2>]]]

Angewendet im Befehlsniveau, listet das im Speicher befindliche Programm ab Zeile <zeile> oder <zeile1> auf dem Display auf, bis zur <zeile2>. Fehlende Angaben werden durch Ende des Programms bzw. Anfang des Programms ersetzt.

LOAD **LOAD <name>**

Lädt ein Programm mit dem Namen <name> in den Speicher. Die neuen Programmteile werden dabei in das sich im Speicher befindliche Programm eingefügt. Bei gleicher Zeilennummer wird die alte Zeile überschrieben. <name> muß eine Zeichenkettenvariable oder -konstante sein, die einer gültigen Filespezifikation entspricht. Möglich ist neben den im Betriebssystem üblichen Geräten (s. Handbuch) das Gerät "B:". Es entspricht der Kassette mit BASICODE-Dateiformat (nicht Programmformat!).

LP. **LP. [<liste der Ausdrücke>]>]**

Druckt Daten auf dem Zeilendrucker, sonst wie PRINT.

NEW **NEW**

Das Programm im Speicher wird ab Zeile 1000 gelöscht, alle Variablen sind ungültig, alle Dateien geschlossen.

O **O. <name>,<access>,<lun>**

Eröffnen einer Datei. <name> ist eine Filespezifikation wie im LOAD-Befehl. <access> ist der Zugriffscode. Zulässig sind:

4 - lesen
6 - Directory lesen (nur für Diskette)
8 - neu anlegen und schreiben
9 - erweitern einer bestehenden Datei

<lun> ist eine logische Kanalnummer. Durch die Besonderheiten des BASICODE-Standards wird das niederwertige Bit ignoriert. 0 und 1, 2 und 3 usw. haben also gleiche Wirkung. Zulässig sind Zahlen von 0 bis 7, es können also maximal 4 Dateien gleichzeitig geöffnet sein. Unmittelbar nach Ausführung des O.-Befehls kann auf Zelle 32 der Status (=0 - o.k., <>0 - Fehler) und im Fehlerfall auf Zelle 35 der Fehlerstatus abgefragt werden.

OSTR$** **OSTR$ (X)

Bringt eine Kette zurück, welche dem Wert von X entspricht.

P. **P. I, J, K**

Schreibt im grafischen Betrieb einen Punkt in der Spalte I und der Zeile J mit der Farbe K. Links oben befindet sich der Punkt 0,0. 0<=I<=319, 0<=J<=191, K=0 entspricht weiß (Punkt löschen), K=1 entspricht schwarz (Punkt setzen).

PEEK **PEEK (X)**
Bringt eine Dezimalzahl als Byte zurück, die auf dem Speicherplatz mit der Adresse X steht.

POKE **POKE I, J**
Speichert ein Byte (dezimaler Wert des Ausdruckes J) auf die Speicherstelle mit der Adresse I.

PRINT **PRINT [<ausdrucksliste>]**
Statt des PRINT kann auch ein Fragezeichen (?) verwendet werden.

R. **R.(I)**
Funktion zum Lesen eines Satzes von der Datei. Die Funktion nimmt den Wert einer Zeichenkette an, die dem nächsten Satz entspricht. I ist die logische Kanalnummer (s. O.-Befehl). Unmittelbar nach der Funktion kann in Speicherzelle 32 der Status (=0 - o.k. <>0 - Fehler), in Zelle 35 der Fehlerstatus gelesen werden. Handelt es sich um eine BASICODE-Datei, ist der Status auf Zelle 6 zu finden. Diese Informationen gehen beim nächsten Ein-Ausgabe-Befehl, also auch bei PRINT oder LP., verloren. Die Zeile PRINT R.(I) ist also nicht sinnvoll, da danach kein Fehlercode mehr ausgewertet werden kann.

RESTORE **RESTORE [<zeile>]**
Setzt den Zeiger, der die DATA-Anweisung liest, auf den Anfang zurück. <zeilenzahl> zeigt auf die DATA-Anweisung, fehlt diese Angabe, wird auf die erste DATA-Anweisung im Programm zurückgesetzt.

RUN **RUN [<zeilenzahl>]**
Vom Befehlsniveau wird ein Programm ab der angegebenen Zeilenzahl bzw. ab Anfang abgearbeitet.

S. **S. I, J, K**
Tonerzeugung. I ist die Dauer in Einheiten von 20 Millisekunden, J die Lautstärke von 0 bis 15. K ist die Tonhöhe von 8 bis 127 und berechnet sich folgendermaßen: 69 ist das mittlere A, also 440 Hz. Für jeden Halbton darüber wird 1 addiert, für jeden Halbton darunter 1 subtrahiert. Das mittlere C entspricht also der Zahl 60. Pitch-Werte unter 8 (sehr tiefer Ton) sind nicht mehr darstellbar und werden intern durch 8 ersetzt. Nach oben gibt es keine Beschränkung.

SAVE **SAVE <name> [<zeile1>-<zeile2>]**
Es wird ein Programm vom Speicher auf Diskette oder Kassette unter dem angegebenen Namen abgespeichert. Die Angaben <zeile1> und <zeile2> haben dabei die gleiche Bedeutung wie beim LIST-Befehl. <name> wird wie beim LOAD-Befehl behandelt. Wichtige Anwendungen sind

SAVE "P:"	Drucken des Programms
SAVE "D:xxx.BC3",1000-	Abspeichern eines BASICODE-Programms auf Diskette
SAVE "D:ROOT.BC3",-999	Abspeichern des BASICODE-Anpassungsprogramms auf Diskette
SAVE "B",-999	Abspeichern des BASICODE-Anpassungsprogramms auf Kassette im BASICODE-Bestandsformat (nicht Programmformat!)

STOP **STOP**
Beendet ein Programm und führt zum Befehlsniveau zurück.

T. **T. I,J,X$**

Schreibt den Text X$ auf den grafischen Bildschirm. Dabei ist I die Adresse im Speicher. Wird angenommen, daß X und Y die Koordinaten des linken oberen Eckpunktes des Textbalkens sind, so berechnet sich I so:

```
I=PEEK(88)+256*PEEK(89)+4Ø*Y+INT(X/8)
```

J gibt an, ob der Text schwarz oder weiß geschrieben wird. Ist J = 0, wird der Text schwarz, ist J = 255, dann weiß geschrieben. Grafikteile, die nicht von Buchstaben berührt werden, bleiben erhalten, auch wenn sie sich im Textbalken befinden.

W. **W.(X$,I)**

Funktion zum Schreiben eines Satzes auf eine Datei. Sie nimmt einen logischen Wert zur Fehlererkennung an (= 0 - o.k, < > 0 - Fehler). X$ ist der zu schreibende Satz, I die logische Kanalnummer (s. O.-Befehl).

Aufruf von ATARI-BASICODE

Vom Betriebssystem ATARI DOS-3 wird der BASIC-Interpreter unter dem Namen

```
BC.CMD
```

verwaltet. Unter diesem Namen wird das Programm geladen und der Befehlsmodus eingestellt. Damit kann der "Taschenrechnerbetrieb" eingestellt werden.

Bei Verwendung der Datasette sind die START- und OPTION-Tasten zu drücken, bei gedrückten Tasten ist der Rechner einzuschalten. Nach dem Piepton ist die Datasette einzuschalten und eine Taste zu drücken. Zuerst wird ein kurzes Ladeprogramm eingelesen, auf dem Bildschirm erscheint die Meldung:

```
LOADING BASICODE...
```

Danach wird der Interpreter gelesen. Dieses File ist identisch mit dem CMD-File der DOS-Version und mit 1200 Baud im TURBO-TAPE-Format aufgezeichnet. Tritt beim Laden ein Fehler auf, meldet sich das Test-Menü.

Verlassen von BASICODE

Die Befehlsfolge:

```
POKE 9,1 <CR>
<RESET>
```

ist die einzige Möglichkeit, um in das Betriebssystem zurückzukehren. <RESET> allein kehrt nicht zum System zurück, sondern zum Befehlsniveau von ATARI-BASICODE.

Eingabe-Editing

Wenn ein falsches Zeichen getippt wurde, kann es mit DELETE gelöscht werden. Eine Zeile kann gelöscht werden, indem SHIFT/DELETE getastet wird. Wird eine Zeile korrigiert, die bereits im Speicher ist, so ist sie einfach unter der gleichen Zeilennummer nochmals richtig zu schreiben. Befindet sich die alte Zeile noch auf dem Bildschirm, kann sie durch die Pfeiltasten angesteuert und danach geändert werden. Bei Eingabe von RETURN wird jeweils die Zeile entgegengenommen, in der sich der Cursor gerade befindet. Die alte Zeile wird damit überschrieben. Soll ein Programm aus dem Speicher entfernt werden, so ist das Kommando NEW zu verwenden, ebenso, wenn ein neues Programm editiert werden soll.

Fehlermeldungen
Wird während der Programmabarbeitung ein Fehler erkannt, so wird das Programm abgebrochen und eine Fehlermeldung ausgegeben. Nun kann dieser Befehl korrigiert werden.

Unterprogramme
Unterprogramme in Assemblersprache sind unter ATARI-BASICODE nicht zugelassen.

Dateiarbeit
Es können unter ATARI-BASICODE nur Textdateien verarbeitet werden. Dazu gibt es den Befehl O. und die Funktionen R., W. und C. Die Verarbeitung von Binärdateien ist nicht vorgesehen.

Diskettenwechsel
Disketten sollen nur im Befehlsniveau nach Drücken der RESET-Taste gewechselt werden. Um ganz sicher zu gehen, sollte man vor dem RESET explizit die Anweisung END eingeben oder einen Fehler provozieren, z. B. durch Eingabe eines unsinnigen Zeichens.

Hinweis:
Unter BASICODE sind die Zeilen 0 bis 999 für systemgebundene Unterprogramme reserviert. Sie sollten auch von Programmierern, die diese Unterprogramme nicht benutzen, freigelassen werden. Eine Reihe von Befehlen verarbeiten nur Zeilen mit Zeilennummern ab 1000.

Wahl der veränderten Hardware bei der Datasette 1010
Zur Verringerung des Hardware-Aufwandes sollte für BASICODE kein separates Kassettengerät angeschlossen werden. Es wurde statt dessen eine relativ leicht zu realisierende Modifikation der Datasette gewählt. Dafür gab es drei Gründe.

Erstens muß in jedem Fall Hardware gebaut werden, die Veränderung der Datasette machte den geringsten Aufwand. Zweitens existiert beim ATARI ohnehin eine übergroße Anzahl von Kabeln und Einzelgeräten. Sie sollte nicht weiter erhöht werden. Drittens ist die programmierte Motorschaltung der Datasette ein wertvolles Hilfsmittel, das so beibehalten werden konnte.

Der Umbau wurde an einer Datasette 1010 durchgeführt. Besitzer anderer Datasetten können bei ihren Geräten analoge Schritte durchführen.

BASICODE wird immer auf beide Kanäle der Kassette aufgezeichnet. Die Aufzeichnung mit beiden Kanälen erhöht die Sicherheit von Kassettenaufzeichnungen im ATARI-Format erheblich. Normal verwendet der ATARI nur den rechten Kanal, um Signale aufzuzeichnen. Der linke Kanal geht direkt zum Audiokanal des Fernsehers oder Monitors und ist für Zusatzinformationen, z. B. Sprache, gedacht. Das zum linken Kanal gehörige Kabel muß vom Tonkopf entfernt werden und beide Tonkopfkanäle phasenrichtig (!) in Reihe (nicht parallel) geschaltet werden. Damit existiert keine übliche Möglichkeit zum Mithören mehr. Um dennoch etwas zu hören, wird der Kollektor des zweiten Vorverstärker-Transistors Q102 über einen Kondensator von 22 nF direkt mit Pin 11 der seriellen Buchse verbunden.

Zur Aufzeichnung von BASICODE sind keine Hardware-Änderungen notwendig, alle Anpassungen sind mittels Software ausgeführt.

Für das Einlesen von BASICODE sind jedoch einige Probleme zu lösen. Die ATARI-Datasetten besitzen einen eingebauten Frequenzdecoder und geben ein bereits decodiertes bitserielles Signal ab. Dieser Decoder muß für die Wiedergabe von BASICODE-Aufzeichnungen außer Betrieb gesetzt werden. Unmittelbar vor dem Signalausgang (Pin 3 der seriellen Buchse) liegt ein Inverter mit offenem Kollektor (Transistor Q103). Die Drahtbrücke an seiner Basis ist aufzutrennen und durch einen einfachen Schalter zu ersetzen. Das noch nicht decodierte Signal wird an der gleichen Stelle (wie oben beschrieben Q102) abgegriffen und auf den Schleifer eines Regelwiderstandes von 68 kOhm gelegt. Das eine feste Ende des Regelwiderstandes liegt auf Masse, das andere an der Basis

von Q103. Der Regler muß bei geöffnetem Schalter so abgeglichen werden, daß beim Einlesen eines BASICODE-Programms Bit 4 des Ports 53775 mit gleicher Häufigkeit 0 und 1 annimmt, bei Kassettenrauschen jedoch immer auf 0 liegt. Das läßt sich sehr einfach mit folgendem Programm durchführen:

```
1Ø POKE 54Ø18,52
2Ø PRINT PEEK(53775)AND16:GOTO 2Ø
```

Wird nun der Schalter geöffnet, ist der Frequenzdecoder unwirksam, und der Recorder kann BASICODE-Kassetten verarbeiten. Dieses Regime wird im folgenden BASICODE-Modus oder einfach "B" genannt. Wird der Schalter geschlossen, so ist wieder der Frequenzdecoder wirksam. Die Widerstandsverhältnisse sind dabei so, daß der nicht dekodierte Pegel, der immer noch an der Basis von Q103 anliegt, unwirksam wird. In diesem Regime kann die Datasette Kassetten im ATARI-Format verarbeiten, es wird ATARI-Modus oder "A" genannt.
Wurde die Hardwareänderung an der Kassette (s. Einleitung) nicht durchgeführt, ergeben sich folgende Einschränkungen:

- Die Befehle CL. und CS. laufen nicht.
- Bei allen Dateibefehlen (LOAD, SAVE, O., C., R., W.) ist das Gerät "B:" (Kassette im BASICODE-Dateiformat) nicht benutzbar.

Alle übrigen Befehle laufen.

Umbau der Datasette XC 12

Der folgende Vorschlag wurde uns inhaltlich als Ergänzung zum vorstehenden Umbau der Datasette 1010 von **W. Koronwski** übermittelt. Es wird ein Hardwarezusatz für die Datasette XC 12 mit 4800 Baud beschrieben, der schon mehrfach benutzt wird. Je nach der Phasenlage des Tonkopfes existieren 2 Varianten:

a) Mit den Schaltkreisen B 555 und DL 000

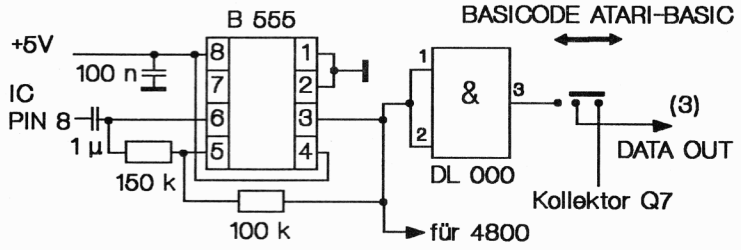


Abbildung 5.20.

b) Mit dem Schaltkreis B 555

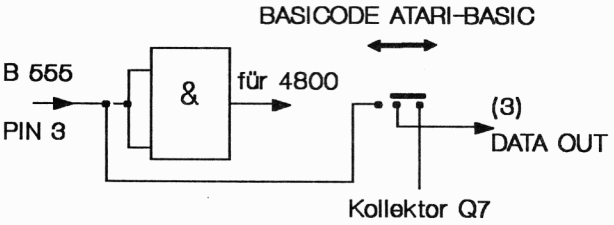


Abbildung 5.21.

Die Erweiterungen hierzu für BASICODE zeigen Abbildungen 5.17. und 5.18. DATA OUT (3) ist auf den jeweiligen Betriebszustand umzuschalten. Bei beiden Varianten braucht kein Leiterzug aufgetrennt zu werden.

Wenn schon der Umbau der Datasette auf 4800 Baud erfolgt ist, ist nur eine kleine Zusatzplatine in die Datasette einzubauen. Ein Abgleich ist nicht notwendig.

Die Änderung am Tonkopf ist wie bei der Datasette 1010 vorzunehmen. Auch der für die Mithörkontrolle benötigte Kondensator von 22 nF ist zu berücksichtigen. Für 4800 Baud ist dann noch eine Leitung zwischen Datasette und Computer zu realisieren, wenn die Audioverbindung für PROCEED benutzt wird. Die Abbildung 5.19. zeigt die Tonkopfanschlüsse.

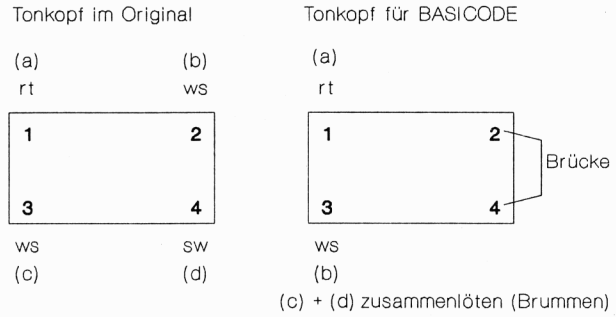


Abbildung 5.22.

5.9. Bascoder für den Sinclair ZX-Spectrum

Verwendbare Rechnertypen

Der hier beschriebene Bascoder eignet sich für jeden zum Sinclair ZX-Spectrum mit 48 oder 80 kByte RAM kompatiblen Rechner mit dem originalen Sinclair-BASIC-Betriebssystem. Neben dem Standard-Kassetten-Interface des ZX-Spectrum sind Microdrives ohne große Probleme nutzbar, Disketteninterfaces je nach Typ jedoch nur mit gewissen Einschränkungen oder Problemen.

Einfache Anleitung

Der Bascoder für den ZX-Spectrum besteht aus zwei Programmteilen:

- BASIC-Programm ("basiCODE 3") und
- Code-Block ("CODE BASICODE 3"CODE).

Beide Teile sind im Standard-Kassettenaufzeichnungsverfahren des ZX-Spectrum einzulesen. Von der Kassette wird der Bascoder folglich mit

```
LOAD"basiCODE 3" <ENTER> oder LOAD"" <ENTER>
```

geladen. Der zweite Programmteil wird also automatisch nachgeladen. Der Teil CODE ist als spezielles Token mit nur einem Tastendruck zu erzeugen. Der Bascoder meldet sich dann mit Titelbild und Versionsnummer.

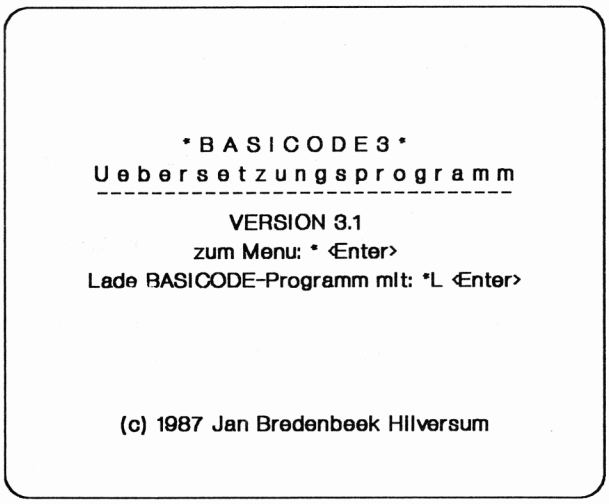


Abbildung 5.23. Startbildschirm

Von hier gelangt man mittels "*" und < ENTER > ins Menü:

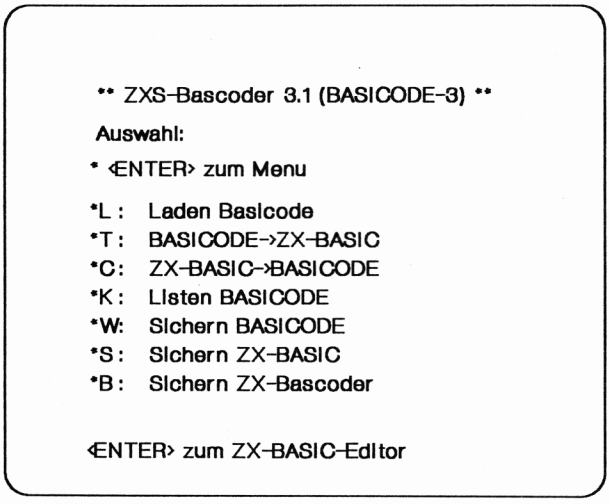


Abbildung 5.24. Menü

Die hier aufgeführten Funktionen seien zunächst kurz erklärt:

*L Laden eines BASICODE-Programms im BASICODE-Aufzeichnungsformat (über die Kassettenschnittstelle),

*T Übersetzen eines im ZX-Spectrum-Speicher befindlichen BASICODE-Programms in ein ZX-Spectrum-BASIC-Programm,
*C Übersetzen eines ZX-Spectrum-BASIC-Programms in ein BASICODE-Programm (in den Arbeitsspeicher),
*W Aufzeichnung eines im ZX-Spectrum befindlichen BASICODE- Programms mit BASICODE-Aufzeichnungsnorm über die Kassettenschnittstelle,
*S Aufzeichnung eines im ZX-Spectrum-Speicher befindlichen BASICODE-Programms im Spectrum-BASIC-Format auf Kassette,
*K Listen des im Speicher befindlichen BASICODE-Programms auf dem Bildschirm,
*B Aufzeichnung des Bascoders im ZX-Spectrum-Aufzeichnungsformat auf Kassette (BASIC- und CODE-Teil).

Bei den Befehlen dürfen Groß- und Kleinbuchstaben verwendet werden. Wenn das Menü nur durch Eingabe von "*" erreicht wurde, kann ein Bascoder-Kommando durch Betätigen der jeweiligen Buchstaben-Taste gestartet werden.

*L Laden von BASICODE-Programmen

Hierdurch befindet sich der Rechner im Wartezustand auf BASICODE-Signale, die an der EAR-Buchse des ZX-Spectrum ankommen. Die richtige Wahl ihres Pegels ist zu erproben. Während des Wartens flackert der Bildschirmrand abwechselnd rot und zyan. Beim Einlesen erscheinen (wie gewohnt blaue und gelbe) Streifen am Bildschirmrand. Wenn gültige Zeichen im BASICODE-Format erkannt werden, so erscheinen sie als Text in der untersten Zeile des Bildschirms. Diese Zeile wird laufend wieder überschrieben.

Das Einlesen wird bei Übertragungsfehlern mit der Fehlermeldung:

```
Tape loading error
```

unterbrochen. Mittels der Break-Taste ist auch eine manuelle Unterbrechung möglich. Dann folgt die Meldung:

```
Break into program
```

Schließlich kann beim Einlesen der im Rechner verfügbare Speicherplatz nicht ausreichen. Dann erscheint die Fehlermeldung:

```
Out of memory
```

Nach erfolgreichem Lesen wird automatisch das Kommando *T des Bascoders aufgerufen, um das eingelesene BASICODE-Programm in ZX-BASIC zu übersetzen.

*T Übersetzen von BASICODE in ZX-BASIC

Dieses Kommando wird selten benötigt, z. B. dann, wenn ein BASICODE-Programm über die Tastatur in den Rechner eingegeben wird bzw. wenn ein mit Lesefehlern behaftetes BASICODE-Programm editiert worden ist. Jede übersetzte Zeile wird nach Aufruf dieses Kommandos in der für den ZX-Spectrum üblichen Weise auf dem Bildschirm aufgelistet.

*C Übersetzen von ZX-BASIC in BASICODE

Dieses Kommando wirkt nur bezüglich Zeilennummern größer als 999, weil nur hier ein BASICODE-Programm zulässig ist. Das BASICODE-Programm wird zusätzlich in einem besonderen Speicherbereich abgelegt. Erst wenn im Laufe der Übersetzung nicht mehr genügend Speicherplatz für beide Programmversionen verfügbar ist, wird nach der Warnausschrift:

```
BASIC PROGRAM OVERWRITTEN
```

das ZX-BASIC-Programm vom BASICODE-Programm stückweise überschrieben.

Beim Übersetzen kontrolliert der Bascoder Zeile für Zeile die Einhaltung der BASICODE-Vorschriften und übersetzt ZX-Token in BASICODE-gerechte ASCII-Zeichenfolgen. Im BASICODE verbotene Befehle des ZX-BASIC-Befehlssatzes werden mit der Meldung:

ERROR

und zugehöriger Zeile sowie einem Fragezeichen hinter dem betreffenden Schlüsselwort angezeigt.

*K Ausgabe eines BASICODE-Programms

Hiermit wird das im Speicher des Rechners befindliche BASICODE-Programm (also ab Zeile 1000) auf dem Bildschirm gelistet. Der Befehl weicht also vom ZX-LIST-Befehl ab, der den gewöhnlichen ZX-BASIC-Programmbereich betrifft.

*W Sichern des BASICODE-Programms

Hierdurch wird an der MIC-Buchse des Rechners ein Signal ausgegeben, das dem BASICODE-Aufzeichnungsformat entspricht und das BASICODE-Programm ab Zeile 1000 überträgt. Der Vorgang kann durch Betätigen der Break-Taste abgebrochen werden.

*S Sichern des ZX-BASIC-Programms

Hiermit kann ein übersetztes BASICODE-Programm im ZX-BASIC- Standard in der ZX-Aufzeichnungsnorm gerettet werden. Das kann auch auf den ZX-Spectrum-typischen Medien, Kassette oder Microdrive, geschehen.

*B Sichern des Bascoders

Hiermit wird der Bascoder in der ZX-typischen Aufzeichnungsart gespeichert. Dies kann auf Kassette oder Microdrive-Cartridge erfolgen. Es wird so also ein Duplikat des Bascoders erzeugt. Dieses Aufzeichnungsformat ist notwendig, weil der ZX-Spectrum nur so den Bascoder einlesen kann.

Massenspeicherbesonderheiten

Für die Arbeit mit BASICODE-Dateien stehen wahlweise Kassetten oder Microdrives zur Auswahl. Für NF = 0 oder 1 wird mit dem BASICODE-Format gearbeitet, für alle übrigen Werte (NF = 2, 4, 6 beim Einlesen, NF = 3, 5, 7 beim Aufzeichnen) kommt das ZX-Format bez. Kassette bzw. Microdrive zur Anwendung. Da der ZX-Spectrum über keine Motorsteuerung angeschlossener Kassettentonbandgeräte verfügt, werden entsprechende Aufforderungen zum Bedienen des Tonbandgerätes vom Bascoder bzw. vom ZX-Betriebssystem auf dem Bildschirm ausgegeben.

Diskettenschnittstellen am ZX-Spectrum arbeiten nach sehr verschiedenen Prinzipien. Dies ist durch ihre jeweilige Einbindung in das Standard-Betriebssystem dieses Rechners bedingt. Nur Diskettenschnittstellen, die exakt das Verhalten von Microdrives simulieren, sind daher vom Bascoder ansprechbar. Es existiert aber eine spezielle Bascoder-Anpassung für das Disketteninterface "β-Disk" der Firma Technology Research. Hierbei wird das komplette BASIC-Programm zusammen mit den Subroutinen des Bascoders als BASIC-File auf Diskette abgelegt. Wenn sich der Teil BASICODE-3-CODE im Speicher befindet, kann daher ein Programm sofort nach dem Laden gestartet werden.

Besondere Befehle

Um eine möglichst genaue Realisierung des BASICODE-Standards auf dem ZX-Spectrum zu erreichen, wurden einige Erweiterungen, teilweise auch Einschränkungen in den BASIC-Editor und den BASIC-Interpreter im BASICODE-Modus des ZX-Spectrum eingebaut.

In BASICODE können beim ZX-Spectrum nun auch Zeilennummern größer als 9999 bis 32767 verwendet werden.

Die String-Befehle wurden etwas modifiziert: Als vom Nutzer definierte Funktionen existieren LEFT$, MID$ und RIGHT$, die deshalb mit dem Vorsatz FN eingegeben werden müssen.

Variablennamen für Strings, Arrays und Schleifenzähler dürfen höchstens zwei Zeichen lang sein. Die Konstruktionen

```
GOTO * n; 2000, 3000, 4000
GOSUB * n; 2000, 3000, 4000
```

ermöglichen, das ON GOTO bzw. ON GOSUB nachzubilden.

Als Steuercodes für die 42-Zeichen-Bildschirmverwaltung dienen:

CODE 24, N	zur Einstellung eines Bildschirmfensters mit N Zeilen (N< =24),
CODE 25	entspricht PRINT Komma,
CODE 26	erlaubt bei Bedarf Scrollen ohne Rückfrage,
CODE 27	schaltet Rückfrage für das Scrollen ein
CODE 16...23,28...31,127	BASICODE-typisch.

Abweichungen gegenüber den Befehlen des ZX-Spectrum-BASIC bestehen vor allem bezüglich:

INPUT, LET, AT, BEEP, PLOT, DRAW, INKEY$, GOTO, GOSUB, EXP(2), DIM, TO, ASC, FOR...NEXT, IF.

5.10. Bascoder für die Schneider-CPC-Rechner

Verwendbare Rechnertypen

Der hier beschriebene Bascoder ist für folgende Rechner der Firma Schneider/Amstrad geeignet:

- CPC464
- CPC664
- CPC6128

Einfache Anleitung

Nach dem Einschalten des Rechners und der BASIC-Editor-Bereitschaftsmeldung ist mittels |TAPE für diskettenorientierte Rechner (CPC 664, 6128) die Kassettenbetriebsart zu wählen. Diese ist standardmäßig beim CPC 464 eingestellt. Das angeschlossene Kassettenlaufwerk mit der Bascoder-Kassette ist dann nach Eingabe von

```
RUN "BASCODER" oder RUN""
```

im Wiedergabe-Modus zu starten. Nach der Anzeige des Ladens von drei BASIC-Blöcken und dem nicht mehr angezeigten Ladevorgang eines vierten (CODE-)Blockes erscheint die Bascoder-Anfangsmeldung:

```
Bascoder für BASICODE-3
(c) 1987 Sjef Simons
(c) 1989 M. van der Meer
```

```
(L) Laden BASICODE
(W) Sichern BASICODE
(B) BASIC- Editor

|HELP zu diesem Menu
```

Damit ist das Bascoder-Programm betriebsbereit. Es besteht aus zwei Teilen: aus dem BASIC-Programm "BASCODER.BAS" und dem CODE-Teil "BC3BIN.BIN". Letzterer wurde vom anfangs geladenen BASIC-Programm automatisch nachgeladen und enthält die wesentlichen Teile zum Übersetzen zwischen CPC-BASIC und BASICODE sowie die typischen BASICODE-Kassettenroutinen.

Ist der Bascoder gestartet, so kann ein BASICODE-Programm über den Kassettenkanal geladen werden. Dazu ist bei Rechnerversionen ohne integriertes Kassettenlaufwerk (CPC664 und CPC6128) das BASICODE-Signal des Kassettenrecorders der Diodenbuchse des Rechners zuzuführen. Dazu wird L betätigt. Danach wird auf dem Bildschirm unten (im MODE 2, also mit 80 Zeichen pro Zeile) nur die Copyright-Meldung:

```
BASICODE-3  (c) 1987 Sjef Simons, (c) 1989 M. van der Meer
```

erscheinen, bis lesbare Zeichen vom Kassettenkanal eintreffen.

Nach dem Vorton werden oben links auf dem Bildschirm beginnend die eingelesenen Zeichen angezeigt. Für die meisten fehlerhaften Zeichen erscheint ein schraffiertes Feld. Diese "Mitschrift" des Einlesevorgangs erfolgt fortlaufend in einem Bildschirmfenster von 21 Zeilen zu je 80 Zeichen so lange, bis das Programm vollständig ist.

Wenn beim Einlesen Fehler auftraten, erscheint eine Fehlermeldung:

```
Checksumerror (Prüfsummenfehler)
```

oder

```
Memory full (Speicher ist voll)
```

Nach dem Einlesen beginnt automatisch die Übersetzung des BASICODE-Programms in den CPC-BASIC-Dialekt. Hierbei erfolgt erneut das Listen des noch nicht übersetzten BASICODE-Programms. Im Rechner wird jedoch die Version im CPC-BASIC abgelegt. Das übersetzte Programm kann nun sowohl mit dem Menü-Punkt W des |HELP-Menüs wieder in BASICODE-Norm als auch im CPC-BASIC-Format gesichert werden (|HELP-Menü BASIC-Editor in üblicher Weise, auch auf Diskette).

Hinweise

Unmittelbar nach dem erstmaligen Laden des Bascoders sollten Sie sich eine Kopie des Bascoders herstellen. Hierzu ist der BASIC-Editor mit B aufzurufen und zunächst der BASIC-Teil mit der Kommandofolge (ggf. nach |TAPE <Enter>)

```
SAVE "BASCODER" <Enter>
```

auf eine neue Kassette abzuspeichern. Dann ist noch der CODE-Teil mit der Kommandofolge

```
SAVE"BC3BIN.BIN",B,&9EØØ,&ØØ88Ø <Enter>
```

auf die Kassette zu retten, und zwar unmittelbar nach den drei Blöcken des BASIC-Teils.

Der Abbruch eines Programms kann durch zweimaliges Drücken von ESC erreicht werden. Mit HELP <Enter> kann dann neu begonnen werden.

Ein Abbruch der Übersetzung kann ebenfalls mit ESC erzwungen werden. Dann erfolgt die Fehlerausschrift:

```
Break in Translation
```

|HELP < Enter > L Laden von BASICODE-Programmen

Das Laden von BASICODE-Programmen erfolgt mit dem |HELP-Menü-Punkt L über den Kassettenkanal, gleichgültig, ob der Rechner eventuell auch über andere Massenspeichermöglichkeiten verfügt.

Treten beim Einlesen bereits auf dem Bildschirm erkennbare Fehler auf, so ist unter Umständen eine Kontrolle der Technik notwendig. Häufig rührt fehlerhaftes Einlesen von falschen Pegeln (Lautstärke) oder schlechter Justage des Magnetkopfes her (siehe Kapitel 4.5.). In jedem Fall sind aber gewisse Korrekturen "von Hand" mittels BASIC-Editor möglich. Beim eventuellen Nachbessern des BASIC-Teils des Bascoders ist zu beachten, daß die Zeile 1000 im Bascoder-BASIC-Teil existiert und damit beim Nachladen von BASICODE-Programmen überschrieben wird. Falls BASICODE-Programme von CPC-typischen Massenspeichern, also im CPC-Kassettenformat aufgezeichnet oder von Diskette geladen werden sollen, so ist einfach zuvor der Bascoder zu starten und dann das Programm aus dem BASIC-Editor heraus mit MERGE und dem entsprechenden Programmnamen in Anführungszeichen dazuzuladen.

Beim Abspeichern bereits in CPC-BASIC übersetzter BASICODE-Programme kann man bei dieser Verwendung des MERGE-Befehls zwecks Sparsamkeit im Umgang mit dem Speichermedium zuvor mit

```
DELETE 1-999 <Enter>
```

den überflüssigen Bascoder-BASIC-Teil löschen und dann mit den üblichen Save-Operationen des CPC das Programm auf Kassette oder Diskette speichern.

```
|HELP<Enter> W  Sichern von BASICODE-Programmen
```

Das Retten von BASICODE-Programmen betrifft nur Zeilen ab 1000 und wird mit dem Menüpunkt W des |HELP-Menus eingeleitet. Zuvor muß der Kassettenrecorder für die Aufnahme gestartet sein. Während der Aufzeichnung erscheint auf dem Bildschirm die Copyright-Meldung sowie :

```
Writing BASICODE - WAIT!
```

Sichern des Bascoders

Das Sichern des Bascoders ist bereits oben beschrieben. Es sind zuvor jedoch eventuell einige Besonderheiten von bestimmten Rechnertypen zu beachten.

Nutzer des CPC 6128 (mit einem angeschlossenen Diskettenlaufwerk) werden künftig den Bascoder von Diskette laden wollen. Nach dem erstmaligen Laden des Bascoders ist dazu der BASIC-Editor mit B aufzurufen und zunächst der BASIC-Teil mit der Kommandofolge:

```
|DISC <Enter>
SAVE "BASCODER" <Enter>
```

auf die Diskette abzuspeichern. Dann ist noch der CODE-Teil mit der Kommandofolge:

```
SAVE"BC3BIN.BIN",B,&9EØØ,&ØØ88Ø <Enter>
```

zu retten. Damit steht Diskettennutzern der Bascoder künftig in der Diskettenbetriebsart (ggf. nach |DISC < Enter >) einfach durch Laden mittels RUN"BASCODER" < Enter > von dieser Diskette zur Verfügung.

Hardware-Besonderheiten

Bei Rechnern des Typs CPC 464 ist herstellerseitig ein Kassettenlaufwerk in das Gehäuse integriert und kein externer Recorder ohne weiteres anschließbar. Um auch hier externe Kassettenrecorder benutzen zu können, ist die Nachrüstung einer Diodenbuchse und die Herausführung der Motorsteuerung des Laufwerks zweckmäßig. Dazu kann auf die erprobte Lösung aus [22] verwiesen werden.

Software-Besonderheiten

Zur Zeit arbeitet das Bascoder-Unterprogramm zum Einlesen des Codes auf der Bildschirmposition des Cursors (aufgerufen mit GOSUB 220) nicht korrekt, wenn das Zeichen an dieser Bildschirmposition invertiert dargestellt ist (z. B. durch Anwendung des Unterprogramms GOSUB 150). Dieser Mangel ist in der pixelorientierten Organisation des Bildwiederholspeichers begründet und wird für spätere Versionen nach Möglichkeit noch behoben.

5.11. Bascoder für 16-Bit-Rechner

Fast alle 16-Bit-Rechner besitzen kein Kassetten-Interface. Andererseits verfügen sie häufig über eine Festplatte. Hierdurch ist ein recht abweichender Betrieb bei BASICODE sinnvoll. Alle notwendigen Arbeits-Programme werden folglich auch nicht von Kassette - in unserem Falle von der Schallplatte - eingelesen, sondern direkt über Disketten verbreitet. Hierfür besteht zum Zeitpunkt des Erscheinens dieses Buches noch keine Möglichkeit. Deshalb tragen die folgenden Aussagen mehr vorläufigen, informierenden Charakter.

Über eine Centronics-Schnittstelle (Parallel-Port) wird mittels eines speziellen Programms und einer zusätzlichen Hardware ein Kassetteninterface erzeugt. Es dient nur zum Einlesen und Schreiben von Programmen und Daten-Files von und auf Kassette. Die andere Arbeit mit BASICODE erfolgt mittels Diskette oder Festplatte. Dementsprechend besteht der Bascoder auch aus zwei getrennten Programmen:

- Programm zum Lesen und Schreiben von BASICODE-Programmen und -Dateien
- Programm zum Arbeiten in BASICODE-Mode.

Das zweite Programm existiert in mehreren Varianten, welche vor allem die jeweilige Grafik-Karte (MDA, CGA, EGA usw. bzw. Hercules) und einige Spezial-Computer betreffen.

Die Beschreibung des Umgangs mit BASICODE erfolgt nun in drei Abschnitten gemäß den o. g. beiden Programmen und der Hardware-Ergänzung.

Das Kassetteninterface

Das Kassetteninterface wird von der Centronics-Schnittstelle abgeleitet. Eine besonders einfache Version zeigt das Bild.

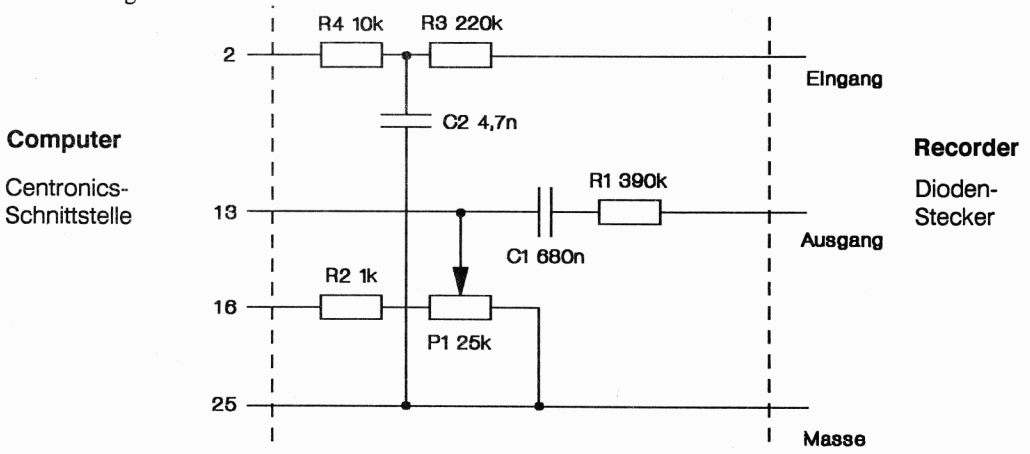


Abbildung 5.25. Einfaches Kassetten-Interface

Die Datenausgabe erfolgt über Pin 2 und gelangt über einen einfachen Tiefpaß aus R4 und C2 sowie einen weiteren Widerstand R3 an den Eingang des Kassetten-Recorders. Da hier am Centronics-Interface ein Spannungshub von etwa 5 Vss zur Verfügung steht, ist dieser Übergang problemlos.

Der Eingang zum Centronics-Interface muß ebenfalls diesen Spannungshub erreichen. Sofern ein Kassetten-Recorder reichlich 1 Veff abgibt, genügt daher der zweite Teil des Interface. Die 390 Ohm dienen der Strombegrenzung für eventuelle Impulse. C1 trennt Gleichpegel ab. Mit dem Potentiometer P1 wird der Ruhepegel von Pin 13 auf etwa 2 Volt eingestellt. Eine genaue Justage dieses Wertes erfolgt über Software.

In mehreren Fällen wird die einfache Variante des Kassetteninterfaces nicht ausreichen. Dann ist gemäß dem untenstehenden Bild ein Verstärker für die Wiedergabe vom Kassettenrecorder einzuschalten.

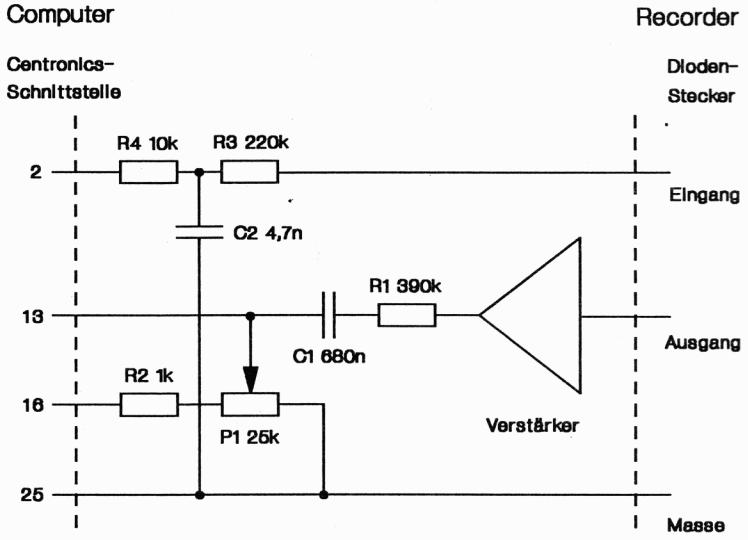


Abbildung 5.26. Kassetten-Interface mit Verstärker

Hierbei ist folgendes zu beachten. Er sollte nicht zu niederohmig (besser strombegrenzt auf etwa 5 mA) sein bzw. auf Spitzenwerte von 5 Vss begrenzen. Ein Problem stellt dabei die Stromversorgung des Verstärkers dar. Optimal ist eine kleine Batterie von 4.5 oder 9 Volt. Sie wird ja nur für die kurzen Zeiten des Einlesens von BASICODE- Programmen oder -Daten benötigt. Diese werden danach ja auf Diskette oder Festplatte abgelegt und von da verwendet.

Weiter kann mit dem Verstärker eine Phasenkorrektur für die Ausgangssignale des Kassetten-Recorders erfolgen. Dadurch kann die Einleserate verbessert werden.

Die Kopplung Kassetteninterface - Diskette

Die Kopplung von Kassetteninterface mit Diskette bzw. Festplatte erfolgt bei den IBM-kompatiblen Rechnern durch das Programm BASICODE.COM. Ihr normaler Aufruf lautet:

```
BASICODE
```

und erfolgt von MS-DOS, d. h. also ohne Benutzung von BASIC. Es erscheint dann ein Menü gemäß:

```
1. Programm einlesen
2. Programm schreiben
3. Daten einlesen
4. Daten schreiben

0=stop    5=trans    6=Option  A=ASCII    L=Lesen    S=Schreiben
```

Die Punkte 1 bis 4 beziehen sich auf den Rechner. Mit 1 bzw. 3 werden also Programme bzw. Daten von der Kassette eingelesen und über entsprechende Zwischenschritte auf Diskette bzw. Festplatte abgelegt. Dieser Vorgang erfolgt im weiteren Verlauf gut durch Anzeigen und Auswahlmöglichkeiten. Er führt im Normalfall dazu, daß die gelesenen Programme und Daten mit einem wählbaren Namen auf Diskette oder der Festplatte abgelegt werden.

Die Punkte 2 und 4 verlangen, daß die entsprechenden Daten bzw. das Programm im Arbeitsspeicher des Rechners stehen. Gegebenenfalls erfolgt das Holen eines Files in den Puffer zusätzlich menugestützt. Dann werden mit diesen Befehlen das Programm bzw. die Daten auf Kassette geschrieben.

Die Punkte 1 bis 4 sind also eindeutig die entscheidenden Teile der Kommunikation zu den anderen BASICODE-Rechnern. Sie werden dementsprechend also nicht besonders häufig benutzt, bilden aber die Voraussetzung für den Übergang zu BASICODE.

Die Option 6 ist wichtig für die Installation und Justage des Kassetteninterface. Hierzu geht man jedoch besser vom noch zu besprechenden INIT-Teil der BASICODE.COM aus.

Der Befehl 0 beendet die Benutzung der BASICODE.COM. Die Befehle 5, A, L, S werden sehr selten notwendig sein und haben die folgende Bedeutung:

- 5 Transformiert ein eingelesenes BASICODE-Programm in die interne BASIC-Darstellung. In der Regel kann dies automatisch nach 1 vorgenommen werden.
- A Zeigt den BASICODE-Puffer in ASCII-Zeichen an. Hier können eventuelle Lesefehler korrigiert werden.
- L Lädt ein Disketten- bzw. Festplatten-File in den BASICODE-Puffer.
- S Schreibt eine Datei in den BASICODE-Puffer.

Installation der BASICODE.COM

Bei der vorangegangenen Besprechung war vorausgesetzt, daß die BASICODE.COM und das Kassetten-Interface bereits optimal eingestellt sind. Um dies zu realisieren, ist es günstig, die BASICODE.COM mit Schalter aufzurufen, weil dann mehr Möglichkeiten bestehen. Der Aufruf lautet:

```
BASICODE/INIT
```

Dann werden bei dem Befehl 6 die folgenden Wahlmöglichkeiten (Optionen) angezeigt:

```
1. Wahl des Port
2. BASICODE-Wahl
3. Kassetten-Justage (*)
4. Speichern der Parameter (*)
E. Fehlerbehandlung
I. Unterbrechung (*)
S. 2400Hz-Signal (*)
```

Ohne den Schalter INIT werden die mit (*) versehenen Punkte nicht zugänglich.

Die Funktion 1 läßt die Auswahl des Druckerinterfaces, also LPT 1 und LPT 2 zu. Auch ein bei den alten IBM-PC vorhandenes Kassetteninterface kann ausgewählt werden.

Mit der Funktion 2 können spezielle BASICODE-Programme und Datenbestände eingelesen werden. Sie ist u. a. für ältere Varianten von BASICODE nützlich.

Funktion E ermöglicht es, beim Einlesen auf Fehler individuell, z. B. mit Abbruch, Übernahme oder Wiederholung zu reagieren.

Die Funktion 4 schreibt die bei der Anpassung als richtig und günstig gefundenen Werte in die BASICODE.COM für die spätere Anwendung.

Die Unterbrechung von Programmen ist unter DOS unterschiedlich festgelegt. Mittels der Funktion I können hierzu Alternativen ausgewählt werden.

Die Funktion S gestattet, einen 2400-Hz-Ton auf Kassette aufzuzeichnen. Er ist günstig für die Justage des Kassetteninterface mit der Funktion 3 und entspricht dem Startton einer Kassettenaufzeichnung (s. Kapitel 4).

Sehr umfangreich sind die Möglichkeiten der Funktion 3. Hier erscheint das folgende Untermenü:

```
1. Anzeige und Wahl des Zeitwertes
2. Automatische Wahl des Zeitwertes
3. Einstellen des Potentiometers
4. Alternative Einstellung zu 3
5. Analyse der gelesenen Pulsbreite
```

Auf eine weitere Untersetzung sei hier verzichtet. Sie erfolgt durch angezeigte Texte unterstützt. Es seien lediglich noch einige Zeitwerte als Richtwerte angegeben:

Typ	CPU	Takt-Frequenz	Zeitgeberwert
Advance 86B	8086	4.77 MHz	25
IBM-AT (kompatible)	80286	8.0 MHz	90
IBM-XT (kompatible)	8088	4.77 MHz	21
IBM-XT (kompatible)	V20	4.77 MHz	22
Olivetti M19	8088	4.77 MHz	21
Olivetti M24	8086	8.0 MHz	39
Olivetti M240	8086	10.0 MHz	49

Arbeiten in BASICODE

In BASICODE-Mode gelangt man auf folgende Weise.

Funktion	Beispiel
Laden von BASICA bzw. GW-BASIC	aus DOS: BASICA
Laden eines Bascoders	LOAD"BAS3"
Zu-Laden eines BASICODE-Programms	MERGE"MOZART"
Starten des Programms	RUN

Es besteht auch die Möglichkeit, den ersten und zweiten Schritt gemäß:

```
BASICA BAS3
```

auszuführen. Eventuell lohnt sich hierfür eine Batch-Datei.

Bei den IBM-PC und Kompatiblen gibt es nun vielfältige Grafik-Karten. Aus diesem Grunde existieren neben BAS3.BAS die Bascoder:

BAS3.BAS	für die CGA-Karte und verwandte Karten
BAS3_OLV.BAS	für die Olivette PC M19, M24, M28, M240, M280
BAS3HERC.BAS	für die Hercules-Karte (hier ist statt BASICA HBASIC nötig)
BAS3EGA.BAS	für die EGA-Karte

Das Retten von BASICODE-Programmen kann auf zwei Arten erfolgen:

- Einschließlich des Bascoder mit SAVE"Name",A
- Ohne Bascoder.

Die zweite Variante ist nützlicher, weil hier das Programm wesentlich kürzer ist und sofort Kompatibilität zur Ausgabe mit anderen Programmen und Ausgabe über BASICODE.COM besteht. Hierbei geht man wie folgt vor:

```
DELETE 1Ø,992
SAVE"Name",A
```

Will man danach im BASICODE weiterarbeiten, genügt es:

```
MERGE"BAS3"
```

aufzurufen und so den Bascoder (statt BAS3 muß bei anderen Karten der entsprechende Name stehen) wieder dazuzuladen.

6. Beispiel-Programme

Hier werden sechs Beispiel-Programme in BASICODE behandelt. Sie sollten ursprünglich auch auf der anliegenden Platte mitgeliefert werden. Im Laufe der Ausarbeitung wurde jedoch die Anzahl der Rechnertypen vergrößert. Dadurch mußte auf diese Programme verzichtet werden. BASICODE-Programme werden jedoch regelmäßig vom Rundfunk der DDR und Radio Hliversum gesendet. Die hier dargestellten Programme sollen die Prinzipien des BASICODE vom Kapitel 2. vertiefen und im Zusammenhang damit die großen Möglichkeiten von BASICODE demonstrieren. Dennoch wurden sie so gestaltet, daß der Anwender sie leicht erweitern und seinen Wünschen anpassen kann. Hierdurch soll im Sinne von Aufgabenstellungen auch der Umgang mit BASICODE geübt werden. Sie sollen zugleich einen anzustrebenden Programmierstil darlegen. Zur besseren Lesbarkeit wird auch hier wieder die Dokumentation mit zusätzlich eingefügtem Space wiedergegeben. BASICODE entfernt sie ja automatisch. Die Programme werden für die Besprechung in ihre Strukturteile zerlegt. Des Verständnisses wegen wird dabei nicht immer die Abfolge der Zeilen eingehalten. Es bleiben aber die Zeilennummern erhalten. Mit den vorgestellten Programmen wird annähernd von einem einfachen Programm "ROMZA" zu immer komplexeren fortgeschritten. Dadurch lassen sich die Erklärungen in erträglichen Grenzen halten. Der Schwerpunkt der Programme betrifft die Gebiete: Textverarbeitung, Grafik und Musik. Numerische Programme wurden ausgespart, da sie in der Regel umfangreicheres mathematisches Wissen verlangen.

6.1. Römische Zahlen

An alten Gebäuden und Denkmälern begegnen uns häufig römische Jahreszahlen. Für den Uneingeweihten ist der Umgang mit ihnen nicht leicht. Ein relativ kurzes BASICODE-Programm kann uns hier behilflich sein. Doch zuvor müssen wir uns ein wenig mit den römischen Zahlen vertraut machen. Zunächst existiert eine spezielle Zahlendarstellung, welche durch die folgenden Symbole gegeben ist:

M=1000 D=500 C=100 L=50 X=10 V=5 I=1

Am einfachsten ist das alte römische System. Hier gab es vier Regeln zur Anwendung dieser Zahlensymbole:

- Die Symbole dürfen nur in der Reihenfolge
  M, D, C, L, X, V, I
  in einer Zahl vorkommen,
- M kann beliebig oft verwendet werden,
- C, X, I dürfen maximal 4mal wiederholt werden,
- D, L, V dürfen nur einmal verwendet werden.

Die arabische Zahl 44 würde hiernach also XXXXIIII lauten. Sie werden aber meist die neuere Schreibweise XLIV finden. Diese entstand mit dem neuen System, welches durch die folgenden Regeln entsteht:

- Es werden sechs neue, zusätzliche Symbole eingeführt:
  CM=900 CD=400 XC=90 XL=40 IX=9 IV=4
- Sie werden in die obige Reihenfolge eingefügt:
  M, CM, D, CD, C, XC, L, XL, X, IX, V, IV, I,
- M kann weiterhin beliebig oft verwendet werden,
- C, X, I dürfen maximal dreimal verwendet werden,
- alle anderen nur einmal
- Für die Doppelsymbole gilt darüber hinaus:
  - vor CM, CD kann nur M stehen,
  - nach CM, CD können die Symbole ab L stehen,
  - vor XC, XL können nur Symbole von M bis C stehen,
  - nach XC, XL können nur Symbole ab V stehen,
  - vor IX, IV können nur Symbole von M bis X stehen,
  - IX, IV sind letzte Symbole

Diese Regeln sind also nicht ganz einfach. Für unser Programm müssen wir beide Richtungen realisieren. Deshalb beginnt das Programm wie folgt:

```
1ØØØ A=25Ø: GOTO 2Ø: REM ## ROMZA ##
1Ø1Ø SR$="ROMZA": HO=INT((HO-LEN(SR$)-6)/2): VE= Ø
1Ø2Ø GOSUB 11Ø: GOSUB 15Ø: PRINT
1Ø3Ø DIM Z(12), Z$(12): F$="Falsche Eingabe"
1Ø4Ø FOR I=Ø TO 12: READ Z(I), Z$(I): NEXT I
1Ø5Ø GOTO 131Ø
```

In Zeile 1000 wird der Stringraum mit A=250 etwas größer als sonst üblich festgelegt. Mit GOTO 20 wird der BASICODE initialisiert. Die Zeile 1010 bereitet die auffällige Anzeige des Programm-Namens vor. Sie wird mit GOSUB 110 oben in der Mitte auf dem Bildschirm vorbereitet und erfolgt mit GOSUB 150 . Dann werden die Felder für die arabischen Zahlen entsprechend den römischen Zeichen vorbereitet. In F$ wird der Fehlerstring abgelegt. Die Zeile 1040 belegt die Felder aus den folgenden DATA-Zeilen. Die Trennung der Programmteile erfolgt meist durch REM-Zeilen:

```
25ØØØ REM------------Daten--------------------
25Ø1Ø DATA 1ØØØ,"M",9ØØ,"CM",5ØØ,"D",4ØØ,"CD",1ØØ
25Ø2Ø DATA "C",9Ø,"XC",5Ø,"L",4Ø,"XL",1Ø,"X",9,"IX"
25Ø3Ø DATA 5,"V",4,"IV",1,"I"
```

Danach erfolgt in Zeile 1050 ein Sprung zu Main (der Hauptroutine mit Eingangsmenü). Sie wurde möglichst weit nach hinten verlegt, damit die Arbeitsprogramme infolge der niedrigeren Zeilennummern schneller arbeiten.

```
13ØØ REM------- Main ---------
131Ø PRINT:INPUT"1 arabisch <-> roemisch 2 =";A: PRINT
132Ø IF (A<1) OR (A>2) THEN PRINT F$: GOTO 131Ø
133Ø ON A GOSUB 1Ø7Ø,117Ø: GOTO 131Ø
```

Mit Eingabe einer "1" erfolgt die Auswahl des Programmteils zur Wandlung von arabischen in römische Ziffern, mit "2" in die umgekehrte Richtung. In Zeile 1320 wird der Fehlerstring ausgegeben. Wird etwas anderes als "1" oder "2" eingegeben, erfolgt erneut die Möglichkeit zur Wahl. In der folgenden Zeile erscheint der Aufruf der Programmteile als Subroutinen. Dies hat mehrere Vorteile für die Programm-Entwicklung, -Änderung und für die Weiterverwendung von Programmteilen:

- Jede derartige Subroutine ist im Sinne der strukturierten Programmierung ein abgeschlossenes einheitliches Ganzes.
- Derartige Subroutinen können bei Erweiterungen oder Änderungen des Programms und sogar anderweitig als fertiger Modul benutzt werden.
- Derartige Subroutinen können einzeln erprobt und so bei Bedarf unabhängig vom Gesamtprogramm geändert werden.
- Für das Hauptprogramm ergibt sich der Vorteil seiner leichten Erprobung. Zunächst können die beiden Subroutinen durch ein RETURN oder kombiniert mit einem REM ersetzt werden, etwa in der Art:

```
1Ø7Ø REM Routine welche arabisch nach römisch wandelt
1Ø8Ø RETURN
```

Das zugehörige wirkliche Programm lautet:

```
1Ø6Ø REM-----------------------
1Ø7Ø N$="": INPUT"Arabische Zahl "; N$:IF N$="" THEN RETURN
1Ø8Ø I=Ø: N=VAL(N$): N$=""
1Ø9Ø IF (N<1) OR (N<>INT(N)) THEN PRINT F$: GOTO 1Ø7Ø
11ØØ IF N>6ØØØ THEN PRINT F$: GOTO 1Ø7Ø
111Ø R=N-Z(I): IF R<Ø THEN I=I+1: GOTO 111Ø
112Ø IF I>12 THEN PRINT F$: GOTO 1Ø7Ø
113Ø N$=N$+Z$(I): N=R
114Ø IF R=Ø THEN PRINT"roemische Zahl = ";N$: GOTO 1Ø7Ø
115Ø GOTO 111Ø
```

Die arabische Zahl wird also als String auf N$ übergeben. Wenn nur ENTER eingegeben wird, entsteht ein Leerstring, und dieses Unterprogramm wird mit RETURN verlassen. In Zeile 1080 wird der String in die Zahl gewandelt. Bei Zahlen kleiner als 1, größer als 6000 (die römische Zahl würde zu lang!) sowie bei nicht ganzzahligen Eingaben wird wieder der Fehlerstring ausgegeben und zur Eingabe zurückgekehrt.

Die Wandlung erfolgt durch schrittweises Abziehen der möglichen römischen Symbole. Dabei wird mit den größten begonnen, und der Rest R darf dabei nicht negativ werden (Zeile 1110). Wenn hierbei die Grenze der möglichen Ziffern überschritten wird, erfolgt wieder die Fehlerausgabe. Gemäß Zeile 1150 wird dies wiederholt, bis in Zeile 1140 R = 0 erkannt wird und so die römische Zahl in N$ aufgebaut wurde.

Mit diesem relativ einfachen Algorithmus sind nicht alle Regeln der römischen Ziffern beachtet. Suchen Sie selbst die Grenzen und ergänzen Sie dementsprechend diese Subroutine. Dabei wird Ihnen die selbständige Struktur nützlich sein.

Die Umwandlung einer römischen Zahl in eine arabische ist einfacher. Die römische Zahl liegt meist richtig vor.

Außerdem genügt hier eine einzige Regel:

- Wenn eine kleinere Zahl vor einer größeren steht, dann ist ihr Wert von der folgenden größeren abzuziehen.

So entsteht die Subroutine:

```
116Ø REM----------------------
117Ø SR$="": INPUT"Roemische Zahl ";SR$
118Ø IF SR$="" THEN RETURN
119Ø A=Ø: B=1: GOSUB 33Ø
12ØØ FOR X=LEN(SR$) TO 1 STEP -1: I=Ø
121Ø   FOR Y=1 TO 7
122Ø    IF MID$(SR$,X,1)=MID$("MDCLXVI",Y,1) THEN I=Y:Y=7
123Ø   NEXT Y
124Ø  IF I=Ø THEN PRINT F$: X=-1: GOTO 127Ø
125Ø  C=Z(I+I-2): IF C<B THEN A=A-C: GOTO 127Ø
126Ø  A=A+C: B=C
127Ø NEXT X
128Ø IF X=Ø THEN PRINT"arabische Zahl =";A
129Ø GOTO 117Ø
```

Die römische Zahl wird in SR$ übergeben. Leerstring (also ENTER) führt wieder aus der Routine heraus. GOSUB 330 wandelt einen eingegebenen String in große Buchstaben um. Es kann also auch viii für VIII eingegeben werden. Von Zeile 1210 bis 1230 wird der X-te Buchstabe des Eingabestrings auf eines der 7 relevanten Grundsymbole "MDCLXVI" untersucht. In I wird der Stellenwert aus der FOR-NEXT-Schleife übernommen. Liegt ein falsches Zeichen vor, so erfolgt die Fehleranzeige. In Zeile 1250 wird die zugehörige arabische Zahl abgelegt. I+I-2 ist notwendig, weil hier nicht die römischen Doppelzeichen benutzt werden. Ist C kleiner als die bereits benutzte größte Zahl in B, so wird der Wert subtrahiert, andernfalls addiert (Zeile 1260). Dieses Prinzip wird auf alle Zeichen des Eingabestrings mit FOR X=... (Zeile 1200) NEXT X (Zeile 1270) angewendet. Die Anzeige erfolgt in Zeile 1280.

Das hier ausführlich Beschriebene ist in Kurzform in den Anmerkungen ab Zeilennummer 30000 abgelegt. Dies ist deshalb sinnvoll, weil meist das Programm ohne Beschreibung verfügbar sein wird. Es ist oft nicht einmal bekannt, was das Programm tut. So wird es pflichtgemäß in diesen Zeilen abgelegt. Es ist eine Frage des Stils, das Wesentliche kurz und dennoch deutlich mitzuteilen.

```
3ØØØØ REM ---------Anmerkungen------------------
3ØØ1Ø REM Es erfolgt eine Umrechnung zwischen
3ØØ2Ø REM arabischen und roemischen Zahlen.
3ØØ3Ø REM Dabei werden die wesentlichen Regeln
3ØØ4Ø REM beruecksichtigt. Es sind aber infolge
3ØØ5Ø REM von Vereinfachungen auch einige
3ØØ6Ø REM ungebraeuchliche Bildungen, insbeson-
3ØØ7Ø REM dere fuer die Eingabe roemischer Zahlen
3ØØ8Ø REM moeglich.
```

Schließlich folgen ab Zeile 32000 noch Hinweise auf den Ursprung des Programms, das Datum usw. Dieser Teil sollte sehr kurz gehalten werden.

```
32ØØØ REM ---------Autorschaft------------------
32Ø1Ø REM H.Voelz; 19.7.89 fuer Buch BASICODE
32Ø2Ø REM Auf IBM-compatiblem Rechner.
```

6.2. Humorvolle Zeilen

Von Carl Reinhardt stammt ein bekanntes Nonsens-Gedicht [3], welches wie folgt beginnt:

> Wenn der Mops mit der Wurst über'n Spucknapf springt
> und der Storch in der Luft den Frosch verschlingt

Dann folgen 13 Variationen, bei denen die Worte Mops, Wurst, Spucknapf, Storch, Luft und Frosch vertauscht werden. Viele, auch große Geister, haben hieran ihre Freude gehabt. So gibt es eine "gesangliche" Anwendung von Karl Valentin. Andererseits ist das Prinzip so einfach und naheliegend, daß man es dem Rechner übertragen sollte. Natürlich muß dabei etwas Neues entstehen. Darüber hinaus sind manuelle Eingriffsmöglichkeiten nützlich. So entstand von mir im Zusammenhang mit dem Rundfunk-Lehrgang [2] die Variante "MOPS". Sie verwendet neun eingebbare Substantive (statt der 6 festen bei Reinhard). Außerdem wurde eine dritte Zeile hinzugefügt (Ich liebe Dreizeiler!). Dieses Programm ist hier nun in BASICODE übertragen und erfreut hoffentlich weiterhin die Nutzer.

Es sind wieder die Standard-Routinen zu Beginn vorhanden, und in Zeile 1040 werden die nötigen Felder dimensioniert. Daran schließt sich für D$() die Belegung mit den drei bestimmten Artikeln an. Sie werden vor allem zur grammatikalischen Richtigkeit der zu konstruierenden Sätze benötigt. In E$() werden dann ihre Dative und in F$() ihre Akkusative abgelegt. Sie hätten natürlich auch in DATA-Zeilen und dann mit READ übernommen werden können. Doch dieser Weg erschien hier übersichtlicher.

```
1000 A=1000: GOTO 20: REM ## MOPS ##
1010 SR$="MOPS": HO=INT((HO-LEN(SR$)-6)/2): VE=0
1020 GOSUB 110: GOSUB 150: PRINT
1030 F$="falsche Eingabe":
1040 DIM A(8), B(8), C$(8), D$(2), E$(2), F$(2)
1050 D$(0)= "DER": D$(1)= "DAS": D$(2)= "DIE"
1060 E$(0)= "DEM": E$(1)= "DEM": E$(2)= "DER"
1070 F$(0)= "DEN": F$(1)= "DAS": F$(2)= "DIE"
```

Damit sind die notwendigen Vorbereitungen getroffen, und nun kann die Eingabe der neun Hauptwörter erfolgen. Hierzu dient eine FOR-NEXT-Schleife.

```
1080 FOR I=0 TO 8
1090   INPUT"Wort mit Artikel";S$
1100   IF MID$(S$,4,1)<>" " THEN PRINT F$: GOTO 1090
1110   SR$=LEFT$(S$,3): GOSUB 330
1120   IF SR$="DER" THEN A(I)=0: GOTO 1160
1130   IF SR$="DAS" THEN A(I)=1: GOTO 1160
1140   IF SR$="DIE" THEN A(I)=2: GOTO 1160
1150   PRINT F$: GOTO 1090
1160   SR$=MID$(S$,4): GOSUB 330: C$(I)=SR$
1170 NEXT I: GOSUB 100
```

Die Eingabe hat in der Form:

> Der HUND, die katze, der Teufel usw.

zu erfolgen.

Dabei ist die Groß-Klein-Schreibung unwesentlich. Mit GOSUB 330 werden beide Wörter ohnehin in Großbuchstaben verwandelt (Zeilen 1110 und 1160). Dies vereinfacht das Programm! Wenn ein Artikel erkannt wird, erhält A(I) den entsprechenden Wert. Anschließend wird das Substantiv in C$(I) abgelegt. Andernfalls erfolgt die Ausgabe des Fehlerstring. Entsprechend der MID$-Bildung in Zeile 1160 ist es notwendig, daß ein Leerzeichen zwischen Artikel und Substantiv steht. Nach Eingabe der neun Substantive wird der Bildschirm gelöscht (GOSUB 100) und es erfolgt der Übergang zur Generierung der Texte.

Es müssen die neun Substantive in unterschiedlicher Reihenfolge ausgewählt werden. Eine gesetzmäßige Permutation ist wegen der Vielzahl von Möglichkeiten (9! = 362880) nicht sinnvoll. Deshalb wird der Zufall mit GOSUB 260 verwendet. Er muß aber so gestaltet werden, daß alle Substantive genau einmal vorkommen. Dies wird durch die Doppelschleife FOR I und FOR J realisiert. Dabei wird in B(I) der zufällige Wert übergeben (Zeile 1200). Außerdem wird in den Zeilen 1210 bis 1230 überprüft, ob dieses Substantiv bereits verwendet wurde. Dann erfolgt eine neue Zufallswahl. Der Prozeß erfolgt iterativ so lange, bis das Ziel erreicht ist:

```
118Ø FOR I=Ø TO 8
119Ø   GOSUB 26Ø: X=INT(8*RV+.5)
12ØØ   C=Ø: IF I=Ø THEN B(I)=X: GOTO 125Ø
121Ø   FOR J=Ø TO I-1
122Ø    IF B(J)=X THEN C=1: J=I
123Ø   NEXT J: IF C=1 THEN 119Ø
124Ø  B(I)=X
125Ø NEXT I
```

Dann kann die Ausgabe erfolgen. Sie ist etwas schwierig zu lesen, denn es sind bis zu drei Felder ineinandergeschachtelt. Dies sei am ersten bestimmten Artikel D$(A(B(0))) in Zeile 1260 erklärt. Der innerste Teil B(0) enthält das Ergebnis der Zufallswahl. A(B(0)) ist das zugehörige Geschlecht und damit D$(A(B(0))) der entsprechende Artikel im Nominativ. Beim folgenden Sustantiv ist alles viel einfacher. Der Rest dürfte nun leichter lesbar sein. Vor allem aber dann, wenn man eine mögliche Textausgabe kennt.

```
WENN DER TEUFEL MIT DER WELT
    UEBER DIE HOELLE SPRINGT
UND DAS ELEND IN DER DISKO
    DAS GELD VERSCHLINGT
DANN DIE LIEBE AUS DER SUENDE
    ALS GOTT ERKLINGT
```

```
126Ø PRINT"WENN ";D$(A(B(Ø)));C$(B(Ø));
127Ø PRINT" MIT ";E$(A(B(1)));C$(B(1))
128Ø PRINT"  UEBER ";F$(A(B(2)));C$(B(2));" SPRINGT"
129Ø PRINT"UND ";D$(A(B(3)));C$(B(3));
13ØØ PRINT" IN ";E$(A(B(4)));C$(B(4))
131Ø PRINT"  ";F$(A(B(5)));C$(B(5));" VERSCHLINGT"
132Ø PRINT"DANN ";D$(A(B(6)));C$(B(6));
133Ø PRINT" AUS ";E$(A(B(7)));C$(B(7))
134Ø PRINT"  ALS";C$(B(8));" ERKLINGT"
135Ø INPUT"Ø weiter 1 Ende ";A$: IF A$="Ø" THEN 118Ø
136Ø GOTO 95Ø
```

Sie erkennen, daß durch die Auswahl der eingegebenen Substantive (keine Plurale verwenden!) automatisch mehr oder weniger witzige Texte entstehen.
Die Anmerkungen und die Autorschaft weisen wieder die wichtigsten Fakten aus.

```
30000 REM -----------Anmerkungen----------
30010 REM Dieses Programm wurde fuer den Rundfunk
30020 REM zum Lehrgang "BASIC 1*1 des Programmierens"
30030 REM auf der Grundlage des Gedichts von Carl
30040 REM Reinhardt "Wenn der Mops mit der Wurst
30050 REM uebern Spucknapf springt" entwickelt.
30060 REM Da es viel Freunde gefunden hat, wurde es
30070 REM hier auf BASICODE angepasst.
30080 REM Beispiel einer Eingabe "der Hund"
32000 REM ----------Autorschaft-------------
32010 REM H. Voelz; 19.7.89; fuer Buch BASICODE
32020 REM Auf XT-compatiblem Rechner
```

6.3. Gedichte in freier Prosa

Nach diesen Vorbereitungen können wir uns dem ersten größeren Programm widmen. Es erzeugt aus einem eingegebenen Wortschatz Zufalls-"Gedichte" in freier Prosa. Mit einer älteren Form von 1983 hatten meine Frau und ich einmal den Einzeiler

"Das rote grün leuchtet blau"

gewonnen. Zusammenhänge hierzu enthält [5]. Dieses Programm erzeugt aber auch Gedichte bis zu 23 Zeilen. Durch eine neue Eingabevariante und die leichten Möglichkeiten des Speicherns von Datensätzen bei BASICODE konnte hier eine neue Qualität erreicht werden. Meines Wissens ist es so sogar das erste in deutsch dazu publizierte Programm. Da es mit vielen Strings umgeht, wird ein Stringraum von 5000 Byte benutzt. Das Programm selbst ist ebenfalls ca. 5 kByte lang. Berücksichtigt man die anderen Speicherbedürfnisse, so müssen also rund 15 kByte frei sein! Dies erkennt man auch an der umfangreichen Definition von Feldern. Ihre Bedeutung ist in den Anmerkungen ab Zeile 30070 erklärt. Ein Teil der Felder wird bei der Vorbereitung des Programms in den Zeilen 1050 bis 1100 belegt.

```
1000 A=5000: GOTO 20: REM ## POEME ##
1010 SR$="POEME": HO=INT((HO-LEN(SR$)-6)/2): VE=0
1020 GOSUB 1110: GOSUB 1500: PRINT: A=50: DIM A(A)
1030 DIM A$(16),B$(3,5),C$(A),D$(A),E$(53),F$(3),G$(3),D(A)
1040 Z=0: U=0: RESTORE
1050 FOR I=0 TO 16: READ A$(I): NEXT I: A=16.4
1060 FOR J=0 TO 5: FOR I=0 TO 3
1070  READ B$(I,J)
1080 NEXT I:NEXT J: B=5.4
1090 FOR I=0 TO 3: READ F$(I): NEXT I
1100 E$(0)="!": E$(1)="?": E$(2)="-"
```

Nun folgt das Hauptmenü. Es unterscheidet 5 Fälle. Mit 1 kann ein Wortschatz eingegeben werden. Er bestimmt wesentlich die Vielfalt und die Spezifik (Genre und Aussagerichtung, wie Weihnachten, Landschaft, Lyrik, Technik usw.). Unter 2 kann ein eventuell schon vorhandener Wortschatz vom Speicher eingelesen werden. Hierdurch ist auch ein schnelles Wechseln zwischen Wortschätzen möglich. Mit 3 können Sie den gerade unter 1 eingegebenen Wortschatz auf das Speichermedium (meist Kassette) retten. Der Punkt 4 schließlich dient zum Erzeugen der Gedichte. Und mit 5 können Sie das Programm beenden.

```
111Ø REM ---- Haupt-Menue -----------
112Ø PRINT: PRINT"Auswahl der Moeglichkeiten": PRINT
113Ø PRINT"1 Text eingeben"
114Ø PRINT"2 Text einlesen"
115Ø PRINT"3 Text speichern"
116Ø PRINT"4 Gedichte erzeugen"
117Ø INPUT"5 Programm beenden";C:_IF C=5 THEN 95Ø
118Ø IF (C<1) OR (C>4) THEN PRINT"## Fehler ##": GOTO 112Ø
119Ø ON C GOSUB 122Ø, 192Ø, 183Ø, 151Ø: GOTO 112Ø
```

Bei Bedarf ließe sich das Programm erweitern. So wäre es unter anderem wünschenswert, den Wortschatz zu editieren oder Wortschätze zu kombinieren. Auch eine Druckerausgabe der Gedichte könnte nützlich sein. Dies soll eine Anregung für Sie sein, das Programm eventuell weiterzuentwickeln. Hier kommt Ihnen dann wieder die klare Strukturierung über GOSUB zugute.

Behandeln wir zunächst die Eingabe des Wortschatzes. Es werden für unsere Gedichte folgende Wortgruppen benötigt:

- Adjektive, wie gut, alt, rot. Je nach Anwendung erhalten sie Endungen: männlich "er", weiblich "e", sächlich "es", plural "en". Dies soll vom Programm automatisch realisiert werden. Deshalb wird die endungslose Grundform gespeichert.
- Substantive, wie Hund, Katze, Frau, Kinder; ihnen muß in A(I) das Geschlecht bzw. Plural zugeordnet werden.
- Bei Verben, wie singt, tanzte, singen, läßt das Programm zwei Wörter zu (z. B. "wird singen"). Deshalb ist oft nur die Gegenwart oder die Vergangenheit möglich. Gegenbeispiele sind aber "wird", "kommt". Es sind also auch die Hilfsverben zugelassen. Für die spätere Nutzung müssen Singular und Plural unterschieden werden.

Um diese Fülle von Möglichkeiten zu vereinfachen, bietet sich die Eingabe von eindeutigen, genau strukturierten Sätzen an. Sie beginnen beim Singular mit den verschiedenen Formen von "ein". Es folgen dann Adjektiv, Substantiv und Verb. Eine Eingabe besteht also aus vier Wörtern getrennt durch je ein Leerzeichen. Beispiele erscheinen zu Beginn als Muster auf dem Bildschirm.

```
12ØØ REM
121Ø REM================EINGABEN====================
1220 PRINT:PRINT"Texteingabe wie folgt:": Print
123Ø PRINT"ein junges Pferd springt"
124Ø PRINT"eine treue Frau liebte"

125Ø PRINT "ein alter Mann raucht"
126Ø PRINT "die bunten Blueten duften"
127Ø PRINT "mit <ENTER> beenden": PRINT
```

Die Eingabe wird auf A$ übergeben und anschließend zerlegt. Der Zähler ist hierbei Z und dient zum Abspeichern der zerlegten Wörter in das ihnen zugehörige Feld. Die Eingabe muß mit RETURN abgeschlossen werden. Z wird dann um das nicht mehr belegte Element zurückgesetzt. Die Zerlegung erkennt unterschiedliche Eingabefehler und weist dann den Eingabestring zurück.

Fehler können vor allem sein:

- Weniger als vier Wörter. Um dies sicher zu erkennen und keine Fehlermeldung mit Programmabbruch hervorzurufen, wird in Zeile 1290 der Eingabestring um einige Leerzeichen verlängert.
- Ein Singular beginnt nicht mit "ein", "eine" oder "einer".
- Ein Plural beginnt nicht mit "die".
- Das Adjektiv hat nicht die richtige Endung.

Es ist deshalb sinnvoll, vor der Übergabe des Strings (mit ENTER) sich noch einmal genau den Text anzusehen. So kann man sich die eventuell notwendige erneute Eingabe ersparen.

```
128Ø A$="": INPUT A$: IF A$="" THEN Z=Z-1: RETURN
129Ø A$=A$+"     ": REM Bestimmung des Geschlechts
13ØØ FOR J=Ø TO 3: L=LEN(A$)
131Ø   FOR I=1 TO L
132Ø    IF MID$(A$,I,1)=" " THEN K=I:I=L
133Ø   NEXT I
134Ø   G$(J)=LEFT$(A$,K-1):A$=RIGHT$(A$,L-K)
135Ø NEXT J
136Ø IF(G$(Ø)="ein")AND(RIGHT$(G$(1),2)="es")THEN 141Ø
137Ø IF(G$(Ø)="eine")AND(RIGHT$(G$(1),1)="e")THEN 142Ø
138Ø IF(G$(Ø)="ein")AND(RIGHT$(G$(1),2)="er")THEN 143Ø
139Ø IF(G$(Ø)="die")AND(RIGHT$(G$(1),2)="en")THEN 144Ø
14ØØ PRINT"falsche Eingabe": GOTO 128Ø
141Ø X=2: D(Z)=Ø: GOTO 146Ø: REM saechlich
142Ø X=1: D(Z)=2: GOTO 146Ø: REM weiblich
143Ø X=2: D(Z)=1: GOTO 146Ø: REM maennlich
144Ø X=2: D(Z)=3: REM           plural
145Ø IF LEFT$(G$(3),1)=" " THEN 14ØØ
146Ø C$(Z)=LEFT$(G$(1),LEN(G$(1))-X)
147Ø D$(Z)=G$(2): E$(Z+3)=G$(3)
148Ø Z=Z+1: IF Z<51 THEN 128Ø
149Ø Z=Z-1: PRINT"Speicher voll": RETURN
```

Schauen wir uns eine Eingabe als Beispiel an:

"Ein junges Pferd springt"

In Zeile 1300 wird mit J = 0 begonnen und dann in Zeile 1320 die Position des ersten Space gesucht und auf K übergeben. Der davor stehende unbestimmte Artikel "ein" wird auf G$(0) übergeben und von diesem Teil einschließlich des Space von A$ abgetrennt. A$ enthält dann nur den Teil "junges Pferd springt" (Die angefügten Space sind echt). Schrittweise werden so die Wörter dem G$(I) zugewiesen. In den Zeilen 1360 bis 1390 erfolgt dann die Analyse auf Geschlecht und Plural. Wenn keine Übereinstimmung im Artikel und der Adjektivendung besteht, erfolgt die Fehlermeldung. Von Zeile 1410 bis 1440 erfolgt die Vorbereitung für die spezifische Abtrennung der Adjektivendung und die Belegung der Feldvariablen D(I). Mit Zeile 1450 wird die Fehlermeldung generiert, falls zu wenig Wörter eingegeben wurden. Die endgültige Ablage der drei Wörter erfolgt schließlich in den Zeilen 1460 bis 1480. Beim Erhöhen der Laufvariablen wird getestet, ob bereits 51 Sätze eingegeben wurden. In diesem Grenzfall erfolgt die Meldung "Speicher voll" und RETURN.
Betrachten wir als nächstes die Generierung der Gedichte. Mit U werden sie numeriert. Diese Zahl wird normal hinter dem auffällig "POEM Nr." angezeigt. V zählt die Zeilenzahl. Sie soll nämlich so begrenzt werden, daß ein Gedicht maximal den Bildschirm mit 25 Zeilen füllt. Mit GOSUB 260 wird immer ine Zufallszahl 0 < = RV < 1 erzeugt. In Zeile 1530 wird dieser Wert mit der Anzahl Z der Eingabezahlen multipliziert und dann daraus das Geschlecht des zugehörigen Substantivs bestimmt und auf X übertragen. Bei der nächsten Zufallsgröße (Zeile 1540) wird entschieden, ob ein Hilfswort A$(A*RV) angezeigt wird oder nicht. Die Grenze 0.6 ist wie alle folgenden Grenzen empirisch festgelegt und kann natürlich auch geändert werden. Mit der nächsten Zufallszahl wird entschieden, ob anschließend ein "Artikel" B$(X,B*RV) ausgegeben wird. Hier wird das Geschlecht "X" benutzt, um eine richtige Auswahl zu treffen. Die nächste Entscheidung betrifft das eventuell auszuwählende

Adjektiv. Hier muß dann mit F$(X) die richtige Endung angefügt werden. Nun erfolgt eine mehrstufige Entscheidung ab Zeile 1660. Ist RV kleiner 0.1, erfolgt ein Sprung nach Zeile 1660. Für 0.1 < = RV < 0.7 erfolgt ein Sprung nach Zeile 1650, wo das richtige, in Zeile 1530 ausgewählte, Substantiv angezeigt wird.

Für noch mögliche Werte < 0.85 oder < 0.9 erfolgen Rücksprünge, so daß Hilfswörter, Artikel und/oder Adjektive zusätzlich eingefügt werden können. Damit dies nicht extrem oft erfolgt und so eine sehr lange Zeile entsteht, ist der Zähler F zur Begrenzung eingefügt. Falls RV > = 0.9 ist, wird schließlich noch die Zeile 1640 durchlaufen, und mit dem Bindestrich entsteht ein zusammengesetztes Substantiv.

```
15ØØ REM========= Generierung ==============
151Ø SR$="POEM Nr."
152Ø U=U+1:PRINT: GOSUB 15Ø :PRINT U: V=Ø: F=-1
153Ø GOSUB 26Ø:F=F+1: ZZ=INT(Z*RV): X=D(ZZ)
154Ø GOSUB 26Ø:IF RV<.6 THEN 156Ø
155Ø GOSUB 26Ø:PRINT A$(A*RV);" ";: REM Hilfswort
156Ø GOSUB 26Ø:IF RV<.5 THEN 158Ø
157Ø GOSUB 26Ø:PRINT B$(X,B*RV);" ";: REM Artikel
158Ø GOSUB 26Ø:IF RV<.35 THEN 16ØØ
159Ø GOSUB 26Ø:PRINT C$(Z*RV);F$(X);" ";: REM Adjektiv
16ØØ GOSUB 26Ø:IF RV<.1 THEN 166Ø
161Ø IF (RV<.7) OR (F>1) THEN 165Ø
162Ø IF RV<.85 THEN F=F+1: GOTO 154Ø
163Ø IF RV<.9 THEN F=F+1: GOTO 158Ø
164Ø GOSUB 26Ø:PRINT D$(Z*RV);"-";: REM Teil-Substantiv
165Ø PRINT D$(ZZ);" ";
```

Mit diesem Teil ist aber erst ein Abschnitt einer Gedichtszeile durchlaufen. Eine erneute Zufallszahl entscheidet nun darüber, ob direkt zu Zeile 1720 gesprungen wird oder ob in Zeile 1670 ein Rücksprung zu Zeile 1530 erfolgt bzw. in den Zeilen 1690 bis 1710 ein grammatisch passendes Verb angehängt wird. Die weiteren Entscheidungen betreffen noch: Anfügen eines Adjektivs (Zeile 1740), Rücksprünge und Erzeugen von einer neuen oder gar einer Leerzeile. In den Zeilen 1750 bzw. 1770 kann das Ende eines Gedichts erreicht werden. Mit der Zeile 1780 kann man sich zwischen einem weiteren Gedicht oder Verlassen der Routine entscheiden.

```
166Ø GOSUB 26Ø:IF RV<.2 THEN 172Ø
167Ø IF (RV<.35) AND (F<1) THEN PRINT", ";:GOTO 153Ø
168Ø GOSUB 26Ø:T=INT((Z+3)*RV): IF T<3 THEN 171Ø
169Ø IF (X<3) AND (D(T-3)=3) THEN 168Ø
17ØØ IF (X=3) AND (D(T-3)<3) THEN 168Ø
171Ø PRINT E$(T);" ";
172Ø GOSUB 26Ø:IF RV<.6 THEN 175Ø
173Ø IF (RV<.7) AND (F<2) THEN 153Ø
174Ø GOSUB 26Ø:PRINT C$(Z*RV)" ";
175Ø PRINT:F=-1: V=V+1: IF V>21 THEN 178Ø
176Ø GOSUB 26Ø:IF RV<.25 THEN F=-1: V=V+1: PRINT
177Ø IF (RV<.95) AND (F<2) AND (V<2Ø) THEN 153Ø
178Ø PRINT: INPUT "Ø=weiter 1=Ende ";C
179Ø IF C=Ø THEN 152Ø
18ØØ RETURN
```

Mit dem Programm macht es zwar keine große Mühe, die Datenfiles zu erzeugen, aber dennoch ist es günstig, sie zu retten. Hierzu dienen die Zeilen 1830 bis 1890. Zunächst ist dem File ein Name zu geben. Dann wird mit NF = 1 und GOSUB 500 die Datei eröffnet. Zum Anfang wird der Wert von Z in einen String gewandelt und dann abgelegt. Dies ist sinnvoll für die Wiedergabe des Files, denn Z bestimmt in der folgenden Schleife seine Länge. Es werden in der FOR-NEXT-Schleife der Reihe nach die Adjektive, Substantive, Verben und das Geschlecht mit GOSUB 300 in String gewandelt übergeben (Zeilen 1860 und 1870). Bei den Adjektiven ist die Erhöhung um 3 notwendig, weil in 0 bis 2 "!","?" und "-" liegen, die mit READ aus den DATA-Zeilen gelesen werden. Dann wird mit GOSUB 580 die Datei geschlossen und mit RETURN diese Subroutine verlassen.

```
181Ø REM
182Ø REM ======= Retten der Daten =======
183Ø INPUT"Name des File =";NF$: NF=1: GOSUB 5ØØ
184Ø SR=Z: GOSUB 3ØØ: GOSUB 56Ø
185Ø FOR I=Ø TO Z
186Ø  SR$=C$(I): GOSUB 56Ø: SR$=D$(I): GOSUB 56Ø
187Ø  SR$=E$(I+3): GOSUB 56Ø: SR=A(I): GOSUB 3ØØ: GOSUB 56Ø
188Ø NEXT I
189Ø GOSUB 58Ø: RETURN
```

In den Zeilen 1920 bis 1980 befinden sich die entgegengesetzten Vorgänge zum Holen von Daten aus einer Datei.

```
19ØØ REM
191Ø REM ======= Lesen von Daten =======
192Ø INPUT"Name des File =";NF$: NF=Ø: GOSUB 5ØØ
193Ø GOSUB 54Ø Z=VAL(IN$)
194Ø FOR I=Ø TO Z
195Ø  GOSUB 54Ø: C$(I)=IN$: GOSUB 54Ø: D$(I)=IN$
196Ø  GOSUB 54Ø: E$(I+3)=IN$: GOSUB 54Ø: A(I)=VAL(IN$)
197Ø NEXT I
198Ø GOSUB 58Ø: RETURN
```

Die folgenden Zeilen enthalten die nötigen Zusatzdaten. Sie erzeugen außerdem beim Start des Programms bereits eine Vorbelegung des Gedichtsfiles, daß ohne Gefahr die Generierung aufgerufen werden kann. Was dann entsteht, ist zwar wenig sinnvoll, aber das Programm läuft technisch fehlerfrei.

```
25ØØØ REM ====== Zusatzdaten =========
25Ø1Ø DATA "also","und","als","oder","auch","sogar"
25Ø2Ø DATA "dann","bald","heute","gestern","so"
25Ø3Ø DATA "weil","dennoch","aber","doch","noch"
25Ø4Ø DATA "morgen","ein","sehr","die","die","kein"
25Ø5Ø DATA "ein","eine","an","ohne","kein","keine"
25Ø6Ø DATA "mit","durch","manch","jene","unter"
25Ø7Ø DATA "dieses","solch","diese","diese","fuer"
25Ø8Ø DATA "mein","ohne","keine","es","er","e","en"
```

Die Anmerkungen fassen in der gewohnten Weise die wesentlichen Aussagen zum Programm zusammen. In diesem Fall ergänzen sie bereits die Aussagen des obigen Textes.

```
30000 REM ----- Anmerkungen --------
30010 REM Es werden Zufallsgedichte in freier Prosa
30020 REM erzeugt. Dabei wird eine kleine Grammatik
30030 REM verwendet. Deshalb ist die streng definierte
30040 REM Eingabe notwendig: Bei Singular ist mit
30050 REM "eine" "eine" bzw. "ein" und bei Plural
30060 REM mit "die" zu beginnen. Nur ein Space!
30070 REM Die Felder bedeuten:
30080 REM A$(16) Sonderwoerter wie: also, und, als ..
30090 REM B$(3,5) Artikel und aehnliches
30100 REM C$(50) Adjektive ohne Endung
30110 REM D$(50) Substantive, D(50) Geschlecht
30120 REM = 0 saechl.; 1 maennl.; 2 weibl.; 3 plural
30130 REM E$(53) Verben und 0="!"; 1="?"; 2="-"
30140 REM F$(3) Endungen es, er, e, en
30150 REM A$ und G$(3) Eingabewandlungen
30160 REM U=Gedicht-Nr.; V=Zeilenzaehler
30170 REM Z=Satzzaehler: A=16.5; B=5.5
30180 REM ZZ fuer Substantiv, X= Geschlecht
32000 REM ------ Autorschaft ---------
32010 REM H. Voelz; 22.7.89; fuer Buch BASICODE
32020 REM Auf IBM-compatiblem
```

6.4. Ein Miniatlas

Legt man sich einen umfangreichen Datensatz von geografischen Koordinaten der Welt an, so sind relativ leicht vielfältige Landkarten damit zu erstellen. Genau dies soll mit dem folgenden Programm an einigen Beispielen demonstriert werden. Der Datensatz enthält "nur" rund 600 typische Koordinaten. Mit ihnen und dem Beispielprogramm können drei unterschiedliche Kartentypen erstellt werden:

- Eine Weltkarte in der Zylinderprojektion,
- ein nach geografischen Längen und Breiten auswählbarer Ausschnitt,
- ein Globusbild vom beliebigen Blickpunkt aus.

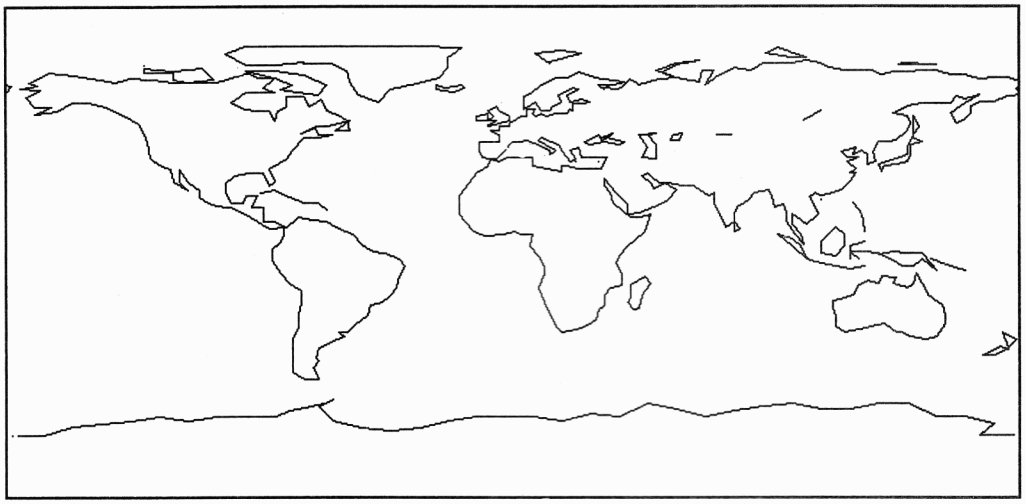


Abbildung 6.27. Weltkarte in Zylinderprojektion

Die letzte Variante wurde von mir erstmalig 1986 [6] vorgestellt und später im Rundfunklehrgang [2] unverändert verwendet. Die jetzige Variante ist stark überarbeitet und speziell auf BASICODE bezogen. Dabei wurde 1988 in einem ähnlichen Programm ein größerer Datensatz von B. Biener verwendet. Hier mußte dieser Datensatz erheblich modifiziert werden.

Das Hauptprogramm bietet nichts wesentlich Neues. Allerdings wurde auf die GOSUB-Routinen im Main verzichtet; denn die ersten beiden Teilprogramme können mit einem gemeinsamen Grundprogramm erfaßt werden. Statt eines Verteilers bleibt so nur noch der Sprung THEN 1320 in Zeile 1030 zum Globus-Teil. Sollten Sie andere Landkarten-Projektionen noch einbauen wollen, wird es sich wiederum als Übung empfehlen, einen GOSUB-Verteiler zu realisieren. Das Programm ist hierfür schon weitgehend vorbereitet.

```
1ØØØ A=1ØØ: GOTO 2Ø: REM ## ATLAS ##
1Ø1Ø SR$="ATLAS": HO=INT((HO-LEN(SR$)-6)/2): VE=Ø:CN=Ø
1Ø2Ø GOSUB 11Ø: GOSUB 15Ø: PRINT: C=(3>2): P1=3.14159
1Ø3Ø INPUT"Ø=Welt; 1=Karte; 2=Globus =";A: IF A=2 THEN 132Ø
```

In der folgenden Zeile werden die Parameter für die Weltkarte festgelegt. Um ein brauchbares Bild zu erhalten, werden Gebiete des Nord- und Südpols ausgeblendet. Die Trennung wurde genau auf den 180. Längengrad gelegt. Dadurch befindet sich die Beringstraße links am Bildrand und der entsprechende Teil Asiens wird am rechten Bildrand abgeschnitten. Mit GOTO 1090 werden die dann folgenden Eingaben für eine Ausschnittskarte übergangen. Für diese Karte wird ein Rechteckausschnitt verwendet, der durch Anfangs- und Endlängen- bzw. Breitengrade festgelegt wird. Es ist zu beachten, daß jener Ausschnitt zu sehen ist, der links mit der Anfangslänge beginnt und rechts mit der Endkoordinate aufhört. Bei den Breitengraden liegt der Anfang südlich. Für Mitteleuropa wären der Reihe nach einzugeben:

Länge: -20 bis 40 und Breite: 30 bis 70

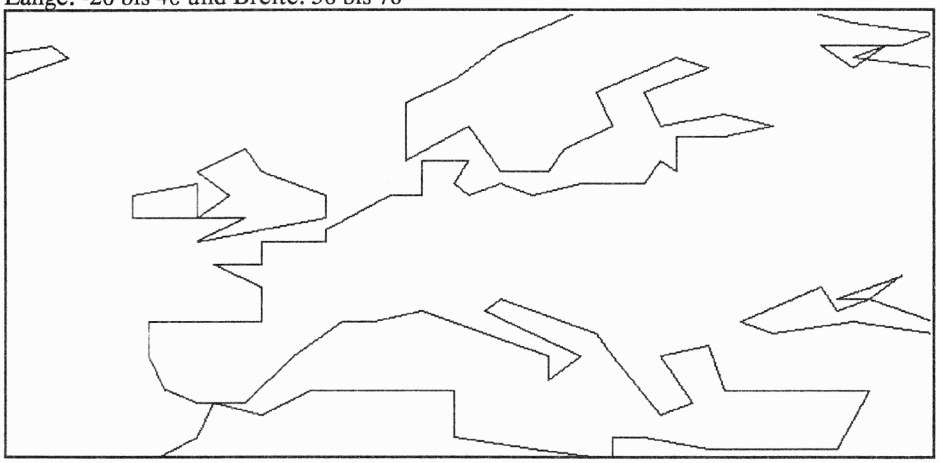


Abbildung 6.28. einfache Karte

```
1Ø4Ø IF A=Ø THEN XX=1/36Ø:YY=-1/2Ø5:XL=18Ø:YL=-1ØØ:GOTO 1Ø9Ø
1Ø5Ø PRINT"Laengen":INPUT"Anfang =";XL:XL=-XL
1Ø6Ø INPUT"Ende =";XX:XX=1/(XX+XL)
1Ø7Ø PRINT"Breite =":INPUT"Anfang =";YY
1Ø8Ø INPUT"Ende =";YL:YY=1/(YY-YL): YL=-YL
```

Das folgende RESTORE setzt den DATA-Zeiger für jeden neuen Start auf den Anfang. Mit GOSUB 600 wird der grafische Betrieb eingeschaltet. Die 1000 in der Schleife ist bewußt so groß gewählt. Es bereitet nämlich bei umfangreichen Datensätzen Schwierigkeiten, die Anzahl zu bestimmen. Deshalb werden hier spezielle Endzahlen angefügt, die das Lesen beenden. Wenn H = 1.5 beträgt, ist das Ende des Datensatzes erreicht. Durch Setzen von I = 1000 und Sprung zum NEXT in Zeile 1180 ist der erste Teil beendet. Zwischendurch geschieht natürlich das Zeichnen der Karte.

```
1Ø9Ø RESTORE: GOSUB 6ØØ
11ØØ FOR I=1 TO 1ØØØ
111Ø    IF C THEN READ H,V:D=V>INT(V):HO=(XL+H)*XX
112Ø    IF C THEN VE=(YL+V)*YY:GOSUB 62Ø:C=D:IF C THEN 111Ø
113Ø    READ H,V: IF H=1.5 THEN I=1ØØØ: GOTO 118Ø
114Ø    C=V>INT(V):H=(XL+H)*XX:V=(YL+V)*YY
115Ø    D=(ABS(HO-H)>.7)OR(ABS(VE-V)>.7)
116Ø    IF D THEN HO=H:VE=V:GOSUB 62Ø:GOTO 118Ø
117Ø    HO=H:VE=V:GOSUB 63Ø
118Ø NEXT I
119Ø INPUT A: GOSUB 11Ø: GOTO 1Ø3Ø
```

Um dies zu verstehen, sind zunächst einige Ausführungen zum Aufbau des Datensatzes notwendig. Er enthält immer zwei zusammengehörige Breiten- und Längenkoordinaten. Sie sind in der Regel ganzzahlig. Dann wird eine Linie vom aktuellen zum neuen Punkt gezogen. In einigen Fällen hat die Länge den "Rest" 0.1, also z. B. 45.1. Dann ist dies der Endpunkt der Linie, und das folgende Punktpaar stellt einen unverbundenen, also neuen Anfangspunkt dar. Einzelne Punkte (kleine

Inseln) haben selbst und zuvor einen nichtganzzahligen Breitenwert. Diese Entscheidung wird auf C bzw. D in den Zeilen 1110 und 1140 erzeugt. Sie wird in 1110, 1120 und 1160 ausgewertet und danach ein Punkt mit GOSUB 620 oder eine Linie mit GOSUB 630 gezeichnet. Die Festlegung in Zeile 1150 hat dagegen einen anderen Grund. Es kann wie bei der Beringstraße eine Linie den Trennschnitt des Bildes überqueren. Dann darf keine Linie gezeichnet werden. Wegen der Bildhöhe und -breite von 1 wurde hier ein Erfahrungswert von 0.7 als Grenze verwendet.

Die restlichen Variablen bewirken eine Anpassung der geografischen Koordinaten auf die Bildkoordinaten von 0 bis 1.

Bevor wir auf den komplizierteren Globusteil eingehen, noch einige Ergänzungen zu Daten. Sie beginnen mit zwei gleichen Werten. Dadurch war ein einfacher und zuverlässiger Start aller Programme möglich. Sie enden mit dreimal 1.5. Dadurch wird grundsätzlich, auch bei Fehlern im Datensatz (Übertragung oder Datenträger) immer ein Ende erreicht.

```
25000 REM---------- Daten ---------
25010 DATA -6,50,-6,50,-2,51,2,52,2,54,-2,56,-3,58
25020 DATA -6,56,-4,54,-6,52,-3,52,-6,50.1,-10,52
25030 DATA -6,52,-6,55,-10,54,-10,52.1,-1,60.1,-6,63.1
25040 DATA 16,76,11,80,20,81,27,80,16,76.1,60,82.1
25050 DATA -18,64,-14,66,-15,67,-21,66,-24,67,-24,65
25060 DATA -18,64.1,-8,37,-9,40,-9,43,-5,43,-2,43
25070 DATA -2,46,-5,48,-2,48,-2,50,2,50,2,51
25080 DATA 6,54,8,54,8,57,11,57,10,55,11,54
25090 DATA 13,55,15,54,18,55,22,55,23,57,24,56
25100 DATA 24,59,27,59,30,60,27,61,23,60,22,63
25110 DATA 26,65,24,66,19,63,20,60,17,58,16,56
25120 DATA 13,56,11,60,7,57,7,62,10,64,13,67
25130 DATA 18,70,22,70,26,72,32,70,35,69,42,68
25140 DATA 38,67,33,67,35,65,37,67,35,66,40,65
25150 DATA 42,67,44,69,46,68,47,67,54,69,60,70
25160 DATA 53,73,60,76,68,77,60,75,55,72,60,70
25170 DATA 68,68,69,73,73,73,70,67,75,71,80,74
25180 DATA 86,73,90,76,95,76,100,76,105,77,110,76
25190 DATA 113,74,118,73,123,73,128,73,131,71,135,72
25200 DATA 140,72,145,73,150,73,155,71,160,71,165,69
25210 DATA 170,70,175,70,180,68,-175,68,-171,67,-173,64
25220 DATA -176,66,180,67,178,65,180,63,175,62,170,60
25230 DATA 165,60,162,58,162,55,160,53,157,51,155,55
25240 DATA 157,57,160,59,165,63,160,61,157,62,155,59
25250 DATA 150,59,145,59,140,58,135,55,140,54,141,51
25260 DATA 140,48,136,44,133,43,130,43,128,39,129,35
25270 DATA 126,34,126,38,124,40,121,38,122,41,117,39
25280 DATA 119,37,123,37,119,35,122,32,120,30,122,30
25290 DATA 120,27,117,23,114,22,110,21,108,22,106,20
25300 DATA 108,16,109,13,105,9,103,11,100,13,100,9
25310 DATA 103,5,104,1,101,5,98,10,98,14,97,17,94,16
25320 DATA 93,20,92,23,87,22,85,19,81,16,80,13
25330 DATA 80,10,77,8,75,13,73,17,73,22,71,21
25340 DATA 68,23,66,25,62,25,57,26,54,26,51,29
25350 DATA 50,30,48,29,51,26,52,24,56,26,60,23
25360 DATA 57,19,52,16,47,14,43,13,43,17,39,22
```

```
25370 DATA 37,25,35,28,35,25,37,22,38,18,41,15
25380 DATA 44,12,47,12,51,13,49,6,46,3,43,0
25390 DATA 40,-4,39,-7,41,-11,41,-16,37,-17,35,-20
25400 DATA 35,-24,33,-26,32,-30,28,-33,25,-34,20,-35
25410 DATA 18,-33,16,-28,14,-23,13,-20,12,-17,13,-14
25420 DATA 14,-11,14,-8,12,-4,9,0,10,5,6,5,4,6
25430 DATA -1,5,-7,4,-11,6,-14,10,-16,13,-16,17
25440 DATA -16,21,-14,24,-10,30,-6,33,-5,36,-2,35
25450 DATA 1,37,5,37,10,37,10,33,15,32,20,31
25460 DATA 20,33,22,33,26,32,31,32,34,32,36,37
25470 DATA 31,37,27,37,26,41,23,40,25,36,23,35
25480 DATA 20,40,19,42,13,45,12,44,18,40,16,38
25490 DATA 16,40,12,42,8,44,5,43,3,43,0,40
25500 DATA -3,36,-6,36,-8,37.1,100,78,105,78,95,82
25510 DATA 91,80,100,78.1,137,75,150,75,138,76,137,75.1
25520 DATA 142,46,142,50,142,54,144,49,142,46,140,41
25530 DATA 140,38,136,36,133,35,130,34,131,31,132,33
25540 DATA 136,34,140,35,141,38,141,42,145,43,142,45.1
25550 DATA 121,23.1,121,18,124,13,125,6.1,105,-5,100,0
25560 DATA 95,5,100,3,103,0,105,-5,110,-7,115,-8
25570 DATA 120,-9,125,-8.1,115,8,110,2,110,-3,115,-4
25580 DATA 118,0,118,5,115,8.1,125,2,120,0,120,-5.1
25590 DATA 123,-5,120,-2,125,-2,130,-3,135,-6,138,-8
25600 DATA 142,-8,145,-5,150,-10,145,-3,140,-3,135,-2
25610 DATA 130,-3.1,148,-5,155,-8,160,-10.1,165,-22.1
25620 DATA 175,-18.1,110,19.1,80,10,80,6,82,7,80,10.1
25630 DATA 49,-12,44,-16,44,-21,43,-23,45,-26,47,-25,48,-20
25640 DATA 51,-15,49,-12.1,33,35.1,25,35.1,14,37.1,9,40.1
25650 DATA 9,42.1,3,39.1,38,47,34,45,36,45,34,44,33,46
25660 DATA 28,43,30,42,35,43,40,42,42,43,36,45
25670 DATA 38,47.1,50,41,48,37,53,36,53,42,50,46
25680 DATA 53,47,47,46,50,41.1,58,44,61,44,62,47
25690 DATA 59,46,58,44.1,74,46,79,46.1,104,52,110,56.1
25700 DATA 143,-11,145,-15,147,-20,152,-24,153,-28,153,-32
25710 DATA 150,-37,146,-38,141,-37,138,-35,136,-35,132,-32
25720 DATA 127,-33,123,-34,118,-35,115,-33,115,-29,114,-25
25730 DATA 114,-22,118,-21,121,-20,123,-16,126,-14
25740 DATA 130,-15,131,-12,136,-13,135,-15,140,-17
25750 DATA 142,-15,143,-11.1,175,-42
25760 DATA 178,-38,173,-35,175,-42,170,-45,166,-45
25770 DATA 172,-41,175,-42.1,147,-43.1,-70,76,-60,76
25780 DATA -55,73,-54,68,-51,63,-44,60,-40,65,-33,67,-23,70
25790 DATA -20,74,-20,78,-15,82,-30,83,-40,83,-50,83
25800 DATA -60,83,-70,83,-80,83,-90,83,-97,80,-90,75
25810 DATA -80,75,-70,76.1,-125,75.1,-100,73.1,-93,74.1
25820 DATA -80,70,-74,67,-78,65,-72,64,-65,63,-62,67,-70,71
25830 DATA -80,73,-90,73,-86,70,-80,70,-84,66,-80,64
25840 DATA -90,64,-95,60,-90,57,-86,56,-82,56,-80,52
25850 DATA -79,56,-76,58,-76,62,-70,62,-68,58,-64,61
25860 DATA -62,56,-57,54,-55,52,-54,48,-59,48,-54,52
```

```
2587Ø DATA -62,5Ø,-66,5Ø,-71,47,-66,49,-65,46,-6Ø,46
2588Ø DATA -65,45,-7Ø,45,-74,4Ø,-76,36,-82,32,-8Ø,27
2589Ø DATA -82,25,-84,3Ø,-89,3Ø,-94,29,-97,26,-97,22
259ØØ DATA -95,18,-91,18,-9Ø,21,-86,21,-88,16,-84,16
2591Ø DATA -84,1Ø,-8Ø,1Ø,-76,8,-72,12,-67,11,-62,1Ø
2592Ø DATA -58,7,-53,5,-5Ø,1,-46,-2,-42,-3,-37,-5
2593Ø DATA -35,-8,-37,-12,-38,-17,-4Ø,-2Ø,-43,-23,-47,-24
2594Ø DATA -48,-28,-52,-32,-55,-35,-58,-35,-56,-37,-6Ø,-39
2595Ø DATA -65,-42,-66,-46,-65,-47,-67,-5Ø,-65,-55,-7Ø,-55
2596Ø DATA -74,-52,-74,-47,-74,-42,-74,-38,-72,-33,-72,-28
2597Ø DATA -71,-23,-71,-18,-75,-15,-78,-11,-81,-6,-81,-1
2598Ø DATA -78,2,-77,5,-77,8,-82,7,-86,1Ø,-9Ø,13
2599Ø DATA -95,16,-97,16,-1Ø2,17,-1Ø6,2Ø,-1Ø6,23,-11Ø,27
26ØØØ DATA -113,32,-113,27,-11Ø,23,-115,27,-116,32,-12Ø,34
261Ø DATA -123,37,-124,43,-125,47,-127,51,-131,54,-138,58
262Ø DATA -144,6Ø,-15Ø,6Ø,-155,56,-163,55,-157,58,-163,58
263Ø DATA -166,62,-161,65,-168,66,-162,67,-165,68,-158,72
264Ø DATA -149,71,-14Ø,69,-135,68,-13Ø,7Ø,-122,69,-115,68
265Ø DATA -1Ø8,68,-1ØØ,68,-95,7Ø,-88,72,-8Ø,7Ø.1
266Ø DATA -125,74,-125,72,-115,72,-115,69,-11Ø,68,-1Ø1,69
267Ø DATA -1Ø5,74,-115,73,-125,74.1,-85,22,-8Ø,23,-75,2Ø
268Ø DATA -7Ø,18,-65,18,-62,15.1,-6Ø,-63,-65,-66,-6Ø,-72
269Ø DATA -5Ø,-75,-4Ø,-76,-3Ø,-75,-2Ø,-72,-1Ø,-7Ø,Ø,-7Ø
27ØØ DATA 1Ø,-7Ø,2Ø,-72,3Ø,-69,4Ø,-7Ø,5Ø,-65,6Ø,-67
271Ø DATA 7Ø,-7Ø,8Ø,-68,9Ø,-66,1ØØ,-65,11Ø,-67,12Ø,-67
272Ø DATA 13Ø,-65,14Ø,-65,15Ø,-7Ø,16Ø,-7Ø,17Ø,-73,165,-78
273Ø DATA 177,-78,-171,-78.1,-169,-78,-16Ø,-78,-15Ø,-74
274Ø DATA -14Ø,-73,-13Ø,-72,-12Ø,-7Ø,-11Ø,-72
275Ø DATA -1ØØ,-7Ø,-9Ø,-7Ø,-8Ø,-7Ø,-75,-68
276Ø DATA -63,-65,-6Ø,-63,1.5,1.5,1.5
```

6.5. Der Globus-Teil

Zum Miniatlas gehört auch ein kleiner in Projektion dargestellter Globus. Je nachdem, welcher Punkt unmittelbar vorne in der Mitte liegt, entsteht ein anderes Bild.

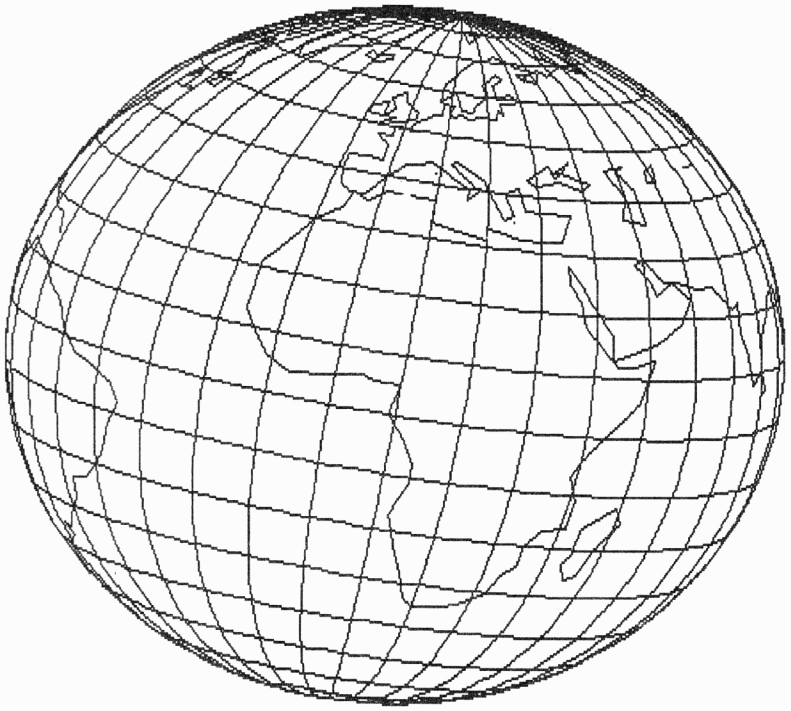


Abbildung 6.29. Globus

Dieser zentrale Punkt wird als Länge und Breite eingegeben. Es sind sogar die Polargebiete möglich. Außerdem steht die Erdachse schief im Weltraum. Auch dies kann mit etwa 23.5 Grad Achsneigung berücksichtigt werden. Schließlich sollen noch zur besseren Veranschaulichung der Kugelgestalt Längen- und Breitenkreise eingezeichnet werden. Ihr Abstand kann günstig zu 45 oder 30 Grad (in obiger Abbildung 10 Grad) wählt werden.

```
131Ø REM------- Globus ----------
132Ø PRINT"Teil Globus":PRINT
133Ø PRINT"alle Winkel in Grad":BM=P1/18Ø
134Ø INPUT"Geografische Laenge=";CW:C=-(CW+9Ø)*BM
135Ø INPUT"Geografische Breite=";AW:A=-AW*BM
136Ø INPUT"Achs-Neigung=";BW:B=-BW*BM
137Ø INPUT"Raster-Abstand=";QW:Q=QW*BM
```

Nun wird der DATA-Zeiger zurückgesetzt und der Grafik-Mode eingeschaltet. Dann können wichtige Kenngrößen für die Projektion berechnet werden. Unter anderem werden der "Radius" R in y-Richtung, R2 in x-Richtung, der Mittelpunkt X0 und Y0 festgelegt. Mit GOSUB 620 wird der erste Punkt gesetzt.

```
138Ø RESTORE:GOSUB 6ØØ:XØ=.5:YØ=.5:R=.45
139Ø S1=SIN(A):S2=SIN(B):S3=SIN(C)
14ØØ C1=COS(A):C2=COS(B):C3=COS(C)
141Ø AX=C2*C3:AY=-C2*S3:AZ=S2
142Ø BX=C1*S3+S1*S2*C3:BY=C1*C3-S1*S2*S3
143Ø BZ=-S1*C2:CX=S1*S3-C1*S2*C3
144Ø CY=S1*C3+C1*S2*S3:CZ=C1*C2
145Ø R2=R*3/4:HO=XØ+R2:VE=YØ:GOSUB 62Ø
```

Jetzt wird der Außenrand der Karte als Kreis gezeichnet. Mit $\pi/50$, also etwa 3.6 Grad, sind die Teilgeraden (GOSUB 630) so fein gestuft (100-Eck), daß sie auch bei der besten Grafikauflösung nicht in Erscheinung treten.

```
146Ø FOR I=P1/5Ø TO 2.Ø1*P1 STEP P1/5Ø
147Ø   HO=XØ+R2*COS(I):VE=YØ+R*SIN(I)
148Ø   GOSUB 63Ø
149Ø NEXT I
```

Hier schließt der zentrale Programmteil an. Bis auf das Flag F3 bezüglich der Sichtbarkeit und kleiner Umrechnungen werden die READ-Werte ähnlich wie bei den Karten bearbeitet.

```
15ØØ REM------- Karte ----------
151Ø F1=Ø:CØ=1
152Ø FOR I=1 TO 1ØØØ
153Ø   READ A,B: IF A<Ø THEN A=36Ø+A
154Ø   F3=Ø: IF B<>INT(B) THEN F3=1
155Ø   IF A=1.5 THEN I=1ØØØ: GOTO 158Ø
156Ø   L=A*BM:P=-B*BM:CL=COS(L):SL=SIN(L)
157Ø   CP=COS(P):SP=SIN(P):GOSUB 121Ø
158Ø NEXT I
```

Alles Entscheidende geschieht jedoch in der Subroutine 1210. Sie besteht aus zwei Teilen einer Koordinatentransformation und dem Zeichnen von Geradenstückchen. Bei der Koordinatentransformation ist es relativ leicht zu entscheiden, ob die Zahlenwerte zur vorderen oder hinteren Halbkugel gehören. IF YY > 0 ist die entsprechende Entscheidung für die hintere Halbkugel. Die Subroutine endet daher nach Setzen weiterer Flag mit RETURN. Andernfalls erfolgen noch die Verschiebungen auf den Globusmittelpunkt und der Ausgleich der "Verzerrung" unseres Bildschirmes mit 3/4. Bei F1 wird (Linienende) der neue Punkt zum alten erklärt.

```
12ØØ REM------- KUGEL->POLAR -----
121Ø Y=R*CP:X=Y*CL:Y=Y*SL:Z=R*SP
122Ø XX=AX*X+AY*Y+AZ*Z:YY=BX*X+BY*Y+BZ*Z
123Ø ZZ=CX*X+CY*Y+CZ*Z:IF YY>Ø THEN F2=Ø:F1=Ø:RETURN
124Ø X2=XØ+XX*3/4:Y2=YØ-ZZ:F2=1
125Ø IF F1=Ø THEN X1=X2:Y1=Y2:F1=1:RETURN
```

Schließlich wird hier eine Gerade oder ein Punkt erzeugt.

```
126Ø REM------- Gerade ---------
127Ø HO=X1:VE=1-Y1:GOSUB 62Ø:IF F4=1 THEN 129Ø
128Ø HO=X2: VE=1-Y2: GOSUB 63Ø
129Ø X1=X2:Y1=Y2:F4=Ø:IF F3=1 THEN F4=1
13ØØ RETURN
```

Nachdem so die Globuskarte gezeichnet wurde, sind die Längen- und Breitenkreise zu ergänzen, natürlich auch nur auf der sichtbaren Hälfte.

```
159Ø REM------ Laengenkreise -----
16ØØ FOR L=Ø TO P1-Q/2 STEP Q
161Ø  F1=Ø:CL=COS(L):SL=SIN(L)
162Ø   FOR P=Ø TO P1+P1+BM STEP 2.5*BM
163Ø    CP=COS(P):SP=SIN(P): GOSUB 121Ø
164Ø   NEXT P
165Ø NEXT L
166Ø REM------ Breitenkreise -----
167Ø FOR P=-P1/2+Q TO PI/2-Q/2 STEP Q
168Ø  F1=Ø:CP=COS(P):SP=SIN(P)
169Ø   FOR L=Ø TO P1+P1+BM STEP 2.5*BM
17ØØ    CL=COS(L):SL=SIN(L): GOSUB 121Ø
171Ø   NEXT L
172Ø NEXT P
```

Abschließend werden auf dem Bild die eingegebenen Parameter angeregt. Der dann folgende Teil verhindert zunächst - bis zu einer Tasteneingabe - die Rückkehr in den Textmode, das Löschen des Bildes und die Rückkehr ins Hauptmenü.

```
173Ø PRINT"Laenge  ="CW:PRINT"Breite  =";AW
174Ø PRINT"Neigung =";BW:PRINT"Raster  =";QW
175Ø INPUT"";A$
176Ø GOSUB 11Ø:GOTO 1Ø3Ø
```

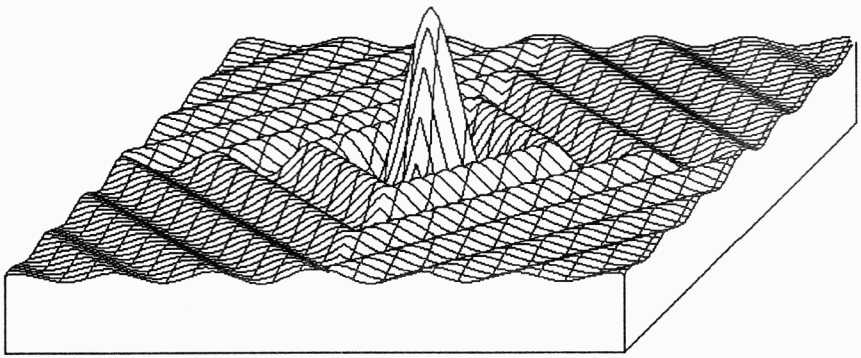
Bleiben nur noch die Anmerkungen darzustellen.

```
3ØØØØ REM -------- Anmerkungen --------------
3ØØ1Ø REM Dies Programm geht auf eine Fassung vom
3ØØ2Ø REM 13.4.85 von mir zurueck, die im BASIC
3ØØ3Ø REM 1*1 des Programmierens verwendet wurde.
3ØØ4Ø REM Jetzt wurde ein groesserer Datensatz
3ØØ5Ø REM von B. Biener, Erfurt verwendet.
3ØØ6Ø REM Die Daten sind wie folgt kodiert:
3ØØ7Ø REM Winkelangaben: geogr. Breite, Laenge
3ØØ8Ø REM Wenn eine Linie endet, dann Breite+.1
3ØØ9Ø REM die letzten 3 Werte muessen 1.5 sein
3Ø1ØØ REM So kann die Karte beliebig verbessert
3Ø11Ø REM werden. Gewuenscht waere ein etwa
3Ø12Ø REM 3mal so grosser Datensatz. Wer
3Ø13Ø REM erzeugt ihn?
3Ø14Ø REM Relativ leicht lassen sich mit dem
3Ø15Ø REM Datensatz vielfaeltige andere Karten
3Ø16Ø REM erzeugen.
32ØØØ REM --------- Autorschaft ---------
```

```
32Ø1Ø REM siehe oben. H. Voelz; 23.7.89
32Ø2Ø REM fuer Buch BASICODE
32Ø3Ø REM IBM-compatibler Rechner
```

6.6. Hiddenline-Darstellung

In Wissenschaft und Technik kommen häufig mathematische Zusammenhänge der Form $z = f(x, y)$ vor. Zu ihrer Veranschaulichung erfolgt dann eine Projektion in die Ebene.



Abbildund 6.30. 3-D-Funktion

Dies läßt sich relativ gut mit der Rechentechnik verwirklichen. Wichtig ist dabei, daß immer etwas im hinteren Teil des Bildes existiert, das durch vordere Teile verdeckt wird, also nicht mehr sichtbar ist. Da man bei der Darstellung Linien benutzt, spricht man von verdeckten Linien oder englisch hidden lines. Derartige Darstellungen sind sehr rechenaufwendig und kosten daher viel Zeit. Ein hochaufgelöstes Bild benötigt zu seiner Darstellung Stunden. Damit ist dies ein Problem, das einerseits an die Grenzen von BASICODE rührt, andererseits werden derartige Abbildungen für viele praktische Anwendungen benötigt. Deshalb kann ein solches Programm bei entsprechender Leistungsfähigkeit schon fast als kommerziell bezeichnet werden. Genau diesem Anspruch soll unser Programm FKT3D gerecht werden. Um dies zu erreichen, wurden einige Besonderheiten realisiert, die wesentlich zu seiner Effektivität beitragen.

Es wird, wie es sich für BASICODE gehört, direkt vom jeweiligen Bildschirmformat (den Pixel-Zahlen) ausgegangen. Dabei wird dieses Format entsprechend dem Zweck gezielt eingeteilt. So ist es erreichbar, daß je Pixel eine Berechnung genügt. Bei vielen Programmen sind es weitaus mehr. Durch diese Lösung kann die Geschwindigkeit um etwa eine Zehnerpotenz gesteigert werden.

Es wird entsprechend den hohen Ansprüchen ein Kreuzraster verwendet. Durch einen speziellen Algorithmus sind hierfür keine zusätzlichen Rechnungen notwendig. Aus schon zuvor berechneten Werten wird eine Geradenkonstruktion abgeleitet.

Zur optimalen Darstellung einer Funktion oder eines gewählten Ausschnittes sind häufig viele Versuche notwendig. Deshalb wurden Hilfsmittel geschaffen, die es erlauben, schnell ein Orientierungsbild zu erhalten. Hierzu wurde ein übersichtliches Bedienerinterface geschaffen, welches leicht eine Datenkorrektur ermöglicht.

Gleich zu Beginn des Programms wurde eine Abweichung vom Standardaufbau eines BASICODE-Programms gewählt. In den Zeilen 1020 bis 1090 wird die darzustellende Funktion abgelegt. Dies geschieht aus folgenden Gründen:

Eine GOSUB-Routine ist um so schneller, je niedriger die beginnende Zeilenzahl ist. Dadurch entfallen eine entsprechende Anzahl von "nutzlosen" Zeilentests.

Die darzustellende Funktion ist je nach Anwendung zu wechseln. Sie ist an dieser Stelle leicht zu finden und damit auszutauschen. Es besteht dabei wenig Gefahr, am eigentlichen Programm etwas zu verändern.

Dieser Weg ist bei den speziellen Eigenschaften von BASICODE gangbar. Von Zeile 1000 wird mit GOTO 20 in den Bascoder gesprungen. Der Rücksprung erfolgt auf Zeile 1010, und von hier wird er auf 1100 weitergeleitet. Dadurch sind die Zeilen 1020 bis 1090 frei. Sie reichen zur Definition selbst extrem komplizierter Funktionen.

```
1000 A=100: GOTO 20: REM ## FKT3D ##
1010 GOTO 1100: REM --- hier ist z = f(x,y) abzulegen ---
1020 Z=ABS(X)+ABS(Y): IF Z<.01 THEN Z=1: RETURN
1030 Z =SIN(Z)/Z
1040 RETURN
1100 HO=INT((HO-LEN(SR$)-6)/2): VE=0: GOSUB 110
1110 SR$="3D-Funktion": GOSUB 150: PRINT
```

Nach dieser Sonderregelung erfolgt die übliche auffällige Darstellung des Programmnamens. Unmittelbar danach wird der Text "bitte etwas warten" ausgegeben. Dies ist notwendig, weil zunächst das Feld für den hidden-Horizont mit Grenzwerten belegt werden muß.

```
1120 REM --- Startwerte ---
1130 XS=-20: XE=20: YS=-20: YE=20
1140 ZS=-.5: ZE=1: F1=3: F2=20: T$="N"
1150 REM --- Hauptroutine ---
1160 REM --- erzeugt das Bild ---
1170 Y2=INT(VG/2): XX=HG-Y2: DIM HI(HG,1), FU(XX,1)
1180 REM --- Feld belegen ---
1190 PRINT "bitte etwas warten"
1200 FOR J=0 TO HG: HI(J,0)=0: HI(J,1)=1: NEXT J
```

Die jetzt folgenden Eingaberoutinen sind so geschrieben, daß der jeweils aktuelle Wert zunächst angezeigt wird. Deswegen erfolgte die Vorbelegung in den Zeilen 1130 und 1140. Wird die Eingabe nur mit ENTER bestätigt, so bleibt der jeweilige Wert erhalten. Wird ein neuer Wert eingegeben, erfolgt seine Übernahme als neuer aktueller Wert. Außerdem werden offensichtlich falsche Eingaben nicht übernommen. Mit der Anzeige "Fehler" wird dann eine neue Eingabe verlangt.

```
1210 REM --- Eingaben ---
1220 PRINT "Startwert von x =";XS;: A$="": INPUT A$
1230 IF A$="" THEN 1250
1240 XS=VAL(A$)
1250 PRINT "Endwert   von x =";XE;: A$="": INPUT A$
1260 IF A$="" THEN 1280
1270 XE=VAL(A$)
1280 IF XE<=XS THEN PRINT "Fehler": GOTO 1220
1290 PRINT "Startwert von y =";YS;: A$="": INPUT A$
1300 IF A$="" THEN 1320
1310 YS=VAL(A$)
1320 PRINT "Endwert   von y =";YE;: A$="": INPUT A$
1330 IF A$="" THEN 1350
1340 YE=VAL(A$)
1350 IF YE<=YS THEN PRINT "Fehler": GOTO 1290
```

Die nun folgenden z-Werte sind nur Richtwerte. Auf Grund der noch zu besprechenden Bildschirmeinteilung werden diese mit Sicherheit dargestellt. In der Regel werden aber auch etwas kleinere und/oder größere Werte im Bild erscheinen.

Die dann folgenden Schrittweiten entsprechen dem Abstand der "horizontalen" und "vertikalen" Gitterlinien. Mit der Änderung des x-Abstandes wird nur das Aussehen des Bildes, nicht aber die Rechenzeit beeinflußt. Die y-Schrittweite hat dagegen ganz wesentlichen Einfluß auf die Rechenzeit. Mit einer großen Schrittweite kann man die Erzeugung eines einfachen Bildes in sehr kurzer Zeit realisieren. Hiermit kann man sich an ein optimales Bild herantasten. Die voreingestellten Werte garantieren auch hier wieder ein für alle Rechner brauchbares Bild.

```
136Ø PRINT "Startwert von z =";ZS;: A$="": INPUT A$
137Ø IF A$="" THEN 139Ø
138Ø ZS=VAL(A$)
139Ø PRINT "Endwert   von z =";ZE;: A$="": INPUT A$
14ØØ IF A$="" THEN 142Ø
141Ø IF ZE<=ZS THEN PRINT "Fehler": GOTO 136Ø
142Ø PRINT "x-Schritt =";F2;: A$="": INPUT A$
143Ø IF A$="" THEN 145Ø
144Ø F2=VAL(A$)
145Ø PRINT "y-Schritt =";F1;: A$="": INPUT A$
146Ø IF A$="" THEN 149Ø
147Ø F1=VAL(A$)
```

Für die nun zu bildenden Hilfswerte sei die Abbildung der gewählten Bildschirmstruktur betrachtet.

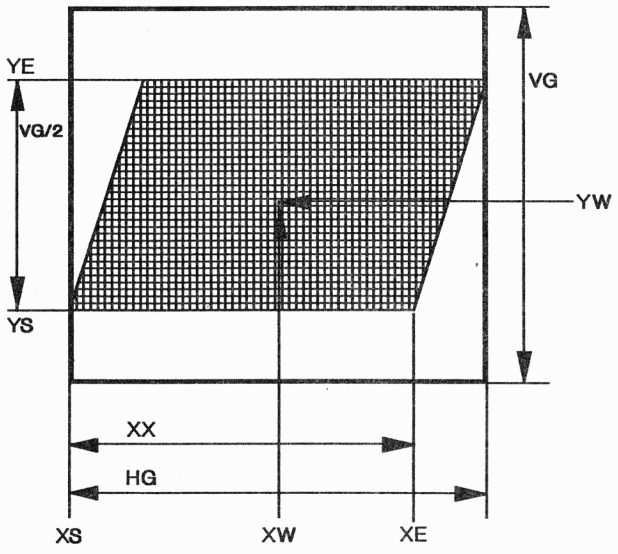


Abbildung 6.31. Grafikschirm

Die Bildschirmstruktur ist durch die Pixelzahlen HG, VG bestimmt. Der Wert VG wird zur Hälfte für die Z-Achse und zur Hälfte für die Y-Achse verwendet. Von den horizontalen Pixeln werden ebenfalls VG/2 für die perspektivische Darstellung genutzt. Der andere, größere Teil dient der X-Achse. Auf diese Weise ergibt sich die gestrichelte Fläche für einen festen Z-Wert. Die darzustellenden Z-Werte ragen aus dieser Quasi-Ebene nach oben bzw. unten heraus.

Die Berechnungsschritte müssen nun in X- und Y-Richtung pixelweise vollzogen werden. Hierzu dienen die ganzzahligen Schleifen. Damit die Endpunkte bei nicht idealer Wahl der Schrittweiten wirklich erreicht werden, wird hier das 0.8fache der Schrittweite sicherheitshalber addiert. Die Schrittweite für die eigentlichen X- und Y-Werte werden in SX bzw. SY festgehalten und immer zu den Werten XW bzw. YW addiert. Mit diesen Werten wird die Funktion über GOSUB 1020 angesprungen. Die HO- und VE-Werte stellen schließlich die abgeleiteten Bildschirmwerte für BASICODE dar. Doch bevor sie verwendet werden, erfolgt noch eine Korrektur mittels MO und MI, falls sie weit außerhalb des Bildschirmes liegen. Hierdurch wird erheblich Rechenzeit bei späteren Routinen erspart.

```
148Ø REM --- Hilfswerte ---
149Ø VZ=Y2/(ZE-ZS): V2=VZ/VG: MO=2: MI=-1: MA=1ØØ: GV=1/VG
15ØØ REM --- Schrittweiten ---
151Ø SX=(XE-XS)/XX: SY=(YE-YS)/Y2: SS=SX*F2
152Ø REM --- Bild darstellen ---
153Ø GOSUB 6ØØ: CN=Ø
154Ø S2=SY*F1: YW=YS-S2
155Ø FOR J=Ø TO Y2+.8*F1 STEP F1: J1=1-J*GV
156Ø  YW=YW+S2: XW=XS-SX: VA=MA
157Ø  IF J>Y2 THEN J=Y2: YW=YE
158Ø    FOR I=Ø TO XX
159Ø      XW=XW+SX: K=I+J: X=XW: Y=YW: HO=K/HG
16ØØ      GOSUB 1Ø2Ø
161Ø      VE=J1+(ZS-Z)*V2
162Ø      IF VE>MO THEN VE=MO
163Ø      IF VE<MI THEN VE=MI
```

Die nun folgende Abfrage dient zwei Zwecken. Zunächst stellt sie fest, ob der erste Punkt in einer Gitterlinie vorliegt. Dann fragt sie ab, ob zwei aufeinanderfolgende VE-Werte einen Abstand größer als zwei Pixel besitzen. Nur in dem kritischen Fall eines zu steilen Y-Anstieges wird der folgende Programmteil durchlaufen. Er verbindet ohne Rechnung (!) die beiden in y-Richtung zu weit entfernten Punkte durch eine Gerade und sorgt so dafür, daß generell auch bei den nur wenigen Berechnungen eine zusammenhängende Gitterlinie entsteht. Hierbei wird bereits die im Feld HI stehende Information über den oberen und unteren Horizont der hidden line abgefragt. Die jeweilige Neubelegung dieses Feldes erfolgt erst anschließend an die Zeilen 1720 bzw. 1730. In diesem Bereich des Programms wird das Feld FU belegt. Es erhält jene Werte, die später für das Querraster verwendet werden.

```
164Ø      IF (VA=MA) OR (ABS(VA-VE)<=GV) THEN 17ØØ
165Ø      S=SGN(VE-VA)/VG: VV=VE
166Ø        FOR VE=VA+S TO VV STEP S
167Ø          IF VE>HI(K,Ø) THEN GOSUB 62Ø
168Ø          IF VE<HI(K,1) THEN GOSUB 62Ø
169Ø        NEXT VE: VE=VV
17ØØ      VA=VE: FU(I,1)=VE
171Ø     IF VE>HI(K,1) THEN FU(I,Ø)=HI(K,1): FU(I,1)=HI(K,1)
172Ø     IF VE>HI(K,Ø) THEN HI(K,Ø)=VE: GOSUB 62Ø
173Ø     IF VE<HI(K,1) THEN HI(K,1)=VE: GOSUB 62Ø
174Ø     IF J=Ø THEN FU(I,Ø)=VE
```

Jetzt folgt die Möglichkeit zur Unterbrechung der Darstellung. Wenn man sich überzeugt, daß das entstehende Bild nicht den Erwartungen entspricht, genügt die Betätigung einer Taste. Dadurch erfolgt sofort der Übergang zur Zeile 2030, wo eine Warteroutine steht. Dadurch kann man sich zunächst noch das Bild in Ruhe ansehen. Erst nach erneutem Betätigen einer Taste wird der endgültige Abbruch und Übergang zur Eingangsroutine vollzogen.

```
175Ø     GOSUB 2ØØ: IF IN<>Ø THEN 2Ø3Ø
176Ø   NEXT I
```

In dem Feld FU liegen immer alle Werte der letzten beiden Zeilen. Falls dabei unsichtbare Pixel existierten, wurde in Zeile 1710 ein entsprechender Horizontwert übergeben. Die Querlinien werden nun zwischen beiden Längslinien im Abstand der X-Schritte ausgewählt und als Geradenstücke zwischen beiden Linien gezeichnet. Ein Fehler kann hier nur dann auftreten, wenn der y-Abstand zu groß gewählt wurde. Aber genau dann ist er auch nicht störend, weil so nur Überblicksbilder erzeugt werden. Nach dem Schreiben des Querrasters werden die alten Werte in dem Feld noch für die nächste Zeile umgespeichert.

```
177Ø   REM --- Querlinien ---
178Ø   FOR I=Ø TO XX+.9*F2 STEP F2
179Ø     IF I>XX THEN I=XX
18ØØ     F=FU(I,Ø)>FU(I,1)
181Ø     IF F THEN HO=(J+I-F1)/HG: VE=FU(I,Ø): GOSUB 62Ø
182Ø     IF F THEN HO=(J+I)/HG: VE=FU(I,1): GOSUB 63Ø
183Ø   NEXT I
184Ø   REM --- Umschreiben der Querlinienwerte ---
185Ø   FOR I=Ø TO XX: FU(I,Ø)=FU(I,1): NEXT I
```

Wie Sie aus dem Beispielbild ersehen, gehört zu jedem Hiddenline-Bild auch noch unten ein Kasten. Er hat mehrere Aufgaben; zunächst wird dadurch das Bild schöner, dann kann dieser Kasten dazu benutzt werden, das Bild zu skalieren. Seine untere waagerechte Linie entspricht dem Abstand (XE-XS). Seine Höhe berücksichtigt (ZE-ZS). Der rechte Teil des Kastens wird erst ganz am Ende geschrieben. Dies hat technische Gründe, zeigt aber auch an, daß der Rechner fertig ist.

```
186Ø   REM ---- unterer Kasten ----
187Ø IF J=Ø THEN HO=Ø:VE=FU(Ø,Ø):GOSUB 62Ø:VE=1-GV:GOSUB 63Ø
188Ø IF J=Ø THEN HO=XX/HG:GOSUB 63Ø: VE=FU(XX,Ø):GOSUB 63Ø
189Ø NEXT J
19ØØ REM ----- rechts unterer Kasten ---
191Ø HO=XX/HG: VE=1-GV: GOSUB 62Ø: HO=1-1/HG:VE=Y2*GV
192Ø GOSUB 63Ø: VE=FU(XX,1): GOSUB 63Ø
193Ø REM --- Bild fertig ---
```

An dieser Stelle ist das eigentliche Bild fertig. Mit GOSUB 210 ist ein "Halt" programmiert. Es kann zum ruhigen Betrachten des Bildes dienen. Bei einigen Rechnern kann es auch genutzt werden, um eine Screenkopie anzufertigen.

Wird dieser Punkt übersprungen, so schreibt die nächste kleine Routine die wesentlichen Bildparameter auf den Schirm, und zwar in die linke obere Ecke. Hier sind fast immer keine Bildpunkte vorhanden. Aber eine Garantie gibt es dafür nicht, deshalb die bewußte Zwischenpause. Da nun erneut wieder GOSUB 210 folgt, gilt das bereits soeben Gesagte.

```
194Ø GOSUB 21Ø: REM Kennwerte schreiben
195Ø SR=F1: GOSUB 3ØØ: A$="Schritte "+SR$+":"
196Ø SR=F2: GOSUB 3ØØ: SR$=A$+SR$: HO=Ø: VE=Ø: GOSUB 65Ø
197Ø SR=XS: GOSUB 3ØØ: A$=SR$+"<x<": SR=XE: GOSUB 3ØØ
```

```
198Ø SR$=A$+SR$: VE=8*GV: GOSUB 65Ø
199Ø SR=YS: GOSUB 3ØØ:A$=SR$+"<y<": SR=YE: GOSUB 3ØØ
2ØØØ SR$=A$+SR$: VE=16*GV: GOSUB 65Ø
2Ø1Ø SR=ZS: GOSUB 3ØØ: A$=SR$+"<z<": SR=ZE: GOSUB 3ØØ
2Ø2Ø SR$=A$+SR$: VE=24*GV: GOSUB 65Ø
2Ø3Ø GOSUB 21Ø: GOSUB 1ØØ
```

Mit der Routine GOSUB 100 wird der Grafik-Mode beendet, es kann ein neues Bild erzeugt bzw. das Programm verlassen werden.

```
2Ø4Ø PRINT "Wie fortsetzen ?"
2Ø5Ø INPUT "E = Ende   N = Neues Bild"; T$:
2Ø6Ø IF (T$<>"E") AND (T$<>"e") THEN T$ = "N": GOTO 119Ø
2Ø7Ø GOTO 95Ø
3ØØØØ REM --------- Hinweise ----------------
3ØØ1Ø REM Programm dient zum 3-dimensionalen Darstellen
3ØØ2Ø REM von Funktionen der Art z = f(x, y)
3ØØ3Ø REM Die Funktion ist den Zeilen 1Ø2Ø bis 1Ø9Ø
3ØØ4Ø REM abzulegen und mit RETURN abzuschliessen.
32ØØØ REM ----------- Autor ------------------
32Ø1Ø REM H. Voelz; 22.8.89; fuer BASICODE-Buch
32Ø2Ø REM ----------------------------------
```

Zum Schluß seien zu Ihrer Anregung noch einige interessante Funktionen genannt.

```
Halbkugel          Z=36-X*X-Y*Y
                   Z=SQR(SGN(SGN(Z)+1)*Z
Spaltfunktion      Z=X*X+Y*Y
                   Z=COS(Z)/(Z+1)
Kantenfunktion     Z=ABS(X*Y)
                   Z=COS(Z)/(Z+1)
Sattelkurve        Z=X*X-Y*Y
```

6.7. MOZART - Ein musikalisches Würfelspiel

Das musikalische Würfelspiel "MOZART - KV 294d" verdeutlicht in recht eigenständiger Weise das "algorithmische Denken" weit vor unserer "digitalen" Zeit. Der Entwurf zu diesem Spiel wird W. A. Mozart zugeschrieben. Es ist jedoch erst 1793, also nach dem Tode Mozarts, bei J. J. Hummel in Berlin-Amsterdam erschienen.

Seinerzeit gehörten zu diesem Spiel zwei Würfel, die Zahlentafeln, der Notenteil mit numerierten Takten, ein Notenheft und natürlich die Spielenden mit ihren Instrumenten.

In der DDR wurde dieses Spiel erstmalig 1986 von Andreas Zierott für einen Computer (KC85/3) umgesetzt. Dieses Maschinenprogramm vollzog den Algorithmus zur Komposition von Walzermelodien auf das Genaueste nach. Diese Leistung kann das vorliegende BASICODE- Programm nicht ganz erfüllen, auf Mehrstimmigkeit mußte verzichtet werden.

```
1ØØØ A=1ØØØ: GOTO 2Ø: REM  **MOZART**
1Ø1Ø REM ** Eroeffnung und Menue **
1Ø2Ø DIMT$(119), WA(12,7), WB(12,7), SV(16Ø), SP(16Ø), SD(16Ø)
1Ø3Ø DIMW$(13)
1Ø4Ø T=1: D=3Ø: TZ=Ø: W=Ø: VO=Ø: SO=2: VT=VE: HT=HO
```

Für den Commodore sind die Sonderbedingungen des Zeichensatzes zu beachten. Zeile 1050 ist eine Möglichkeit. Die Codierung von "A" zeigt an, ob ein Commodore vorliegt. Die Variable NC kann zu jeder Zeit im Programm abgefragt werden (Zeile 2720). Es gilt:
NC = 0 wenn ASC ("A") < > 65.

Für den Commodore sind hier die Sonderbedingungen des Zeichensatzes zu beachten.

```
1Ø5Ø NC=(ASC("A")=65): REM Not Commodore-64
1Ø6Ø RESTORE
1Ø7Ø FOR VE=1 TO 5: GOSUB 293Ø: GOSUB 15Ø: NEXT VE
1Ø8Ø W=Ø
1Ø9Ø FOR VE=9 TO 19 STEP 2: GOSUB 293Ø: PRINT SR$;: NEXT VE
11ØØ GOSUB 12Ø: HE=HO: VE=9: GOSUB 11Ø: PRINT D;
111Ø VE=11: GOSUB 11Ø: PRINT SO;
112Ø VE=22: GOSUB 293Ø: GOSUB 15Ø
113Ø FOR W=7 TO 13: READ W$(W): NEXT W
```

Hier beginnt das Einlesen der Noten aus den DATA-Zeilen. Gegenüber der Textvorlage des KV 294d, in welcher 176 Takte Verwendung fanden, sind es hier nur 119. Alle Wiederholungen, auch jene, die durch die "Einstimmigkeit" entstanden, wurden entfernt.

Die Noten werden hierbei noch nicht entschlüsselt. Die Felder müßten für SV, SP und SD dann in zwei Dimensionen (118,8) angelegt werden. Ein derartiges Vorgehen würde nämlich fast 11000 Byte erfordern.

```
114Ø REM
115Ø REM ** Noten und Takte lesen **
116Ø REM
117Ø FOR T=1 TO 119: READ T$(T): NEXT T
118Ø FORZ=2 TO 12: FOR WT=1 TO 7
119Ø   READ WA(Z,WT): READ WB(Z,WT)
12ØØ NEXT WT: NEXT Z
```

Das Programm wurde so gestaltet, daß der Betrachter sich aktiv in das Spiel einbeziehen lassen kann. Zu diesem Zweck können verschiedene Funktionen und Parameter eingestellt werden.

```
15ØØ REM
151Ø REM ** Funktionsauswahl **
152Ø REM
153Ø X=7: GOSUB 313Ø: IF (IN$<"1") OR (IN$>"6") THEN 153Ø
154Ø X=IN-48: ON X GOTO 353Ø, 363Ø, 373Ø, 383Ø, 2Ø3Ø, 95Ø
```

Das Hauptprogramm ist eine unterbrechbare Endlosschleife. Im Programmablauf sind Zufall und Gesetzmäßigkeit kombiniert. Jeweils 24 Takte werden durch den Zufallsgenerator ausgewählt. Über die Zahlentafeln erfolgt der determinierte Ablaufteil. Die ersten 2*8 Takte werden der Tabelle WA entnommen, die letzten 8 Takte liefert die Tabelle WB. Die Tabellen nehmen in unserem Beispiel aber nur jeweils die Takte 1-7 eines Walzerteils auf, der Schlußtakt ist innerhalb einer Zahlentafel gleich und wird gesondert angefügt.

Für einen Walzer werden zunächst alle zugehörigen Sound-Parameter berechnet und in den Variabelfeldern SP(TZ), SD(TZ) und SV(TZ) abgelegt. Damit entfällt nicht nur erhebliche Rechenzeit zwischen den Tönen, sondern es existiert auch ein Puffer für Takteingaben des Spielers.

```
2ØØØ REM
2Ø1Ø REM ** 24 Takte Mozart-Walzer **
2Ø2Ø REM
2Ø3Ø GOSUB 3Ø3Ø: X=8: GOSUB 323Ø
```

```
204Ø FOR I=1 TO 2
205Ø   FOR WA=1 TO 7
206Ø     GOSUB 253Ø: T=WA(Z,WA): GOSUB 271Ø
207Ø   NEXT WA
208Ø   SD=D/2: SP=67: SV=12: GOSUB 263Ø
209Ø NEXT I
21ØØ FOR WB=1 TO 7
211Ø  GOSUB 253Ø: T=WB(Z,WB): GOSUB 271Ø
212Ø NEXT WB
213Ø T$(Ø)="MA5": T=Ø: GOSUB 271Ø
214Ø SV(TZ+1)=-1: TZ=Ø
```

Über diese Programmschleife laufen alle Tonausgaben. Abbruch erfolgt durch einen Tastendruck oder durch das Endekennzeichen (SV < 0).

```
215Ø REM
216Ø REM ** Notenfeld vorspielen **
217Ø REM
218Ø X=9: GOSUB 323Ø: SR$=W$(13): GOSUB 333Ø
219Ø FOR TZ=1 TO 16Ø
22ØØ   SV=SV(TZ): SP=SP(TZ): SD=SD(TZ)
221Ø   IF SV=12 THEN GOSUB 4ØØ: GOTO 225Ø
222Ø   GOSUB 2ØØ
223Ø   IF (IN>Ø) OR (SV<Ø) THEN TZ=16Ø: GOTO 225Ø
224Ø   GOSUB 4ØØ
225Ø NEXT TZ
226Ø TZ=Ø: IF SV=-1 THEN 23Ø
227Ø GOTO 153Ø
```

Die Spielanleitung von Mozart fordert die Verwendung von zwei Würfeln. Damit lassen sich die Zufallszahlen von 2-12 erzeugen. So sind durch die 11 möglichen Tabellenzugriffe je Takt schon mehr als 45 Billiarden Variationen für nur 16 Takte eines Walzers möglich, und das ohne den Wiederholungsteil.

```
25ØØ REM
251Ø REM ** Zufallszahl 2-12 **
252Ø REM
253Ø GOSUB 26Ø: Z=INT(RV*11)+2
254Ø RETURN
```

Jedes erzeugte Notenfeld kann unterschiedliche Länge besitzen. Die Anzahl der Noten je Takt schwankt nämlich zwischen 3 (3/8 Noten) und 9 (Triller).

```
26ØØ REM
261Ø REM ** Notenfeld aufbauen **
262Ø REM
263Ø TZ=TZ+1: SD(TZ)=SD: SV(TZ)=SV: SP(TZ)=SP+12-SO*12
264Ø RETURN
27ØØ REM
271Ø FOR L=1 TO LEN(T$(T))
272Ø   HA=ASC(MID$(T$(T),L,1)): IF NC THEN 274Ø
273Ø   HA=HA+32: IF HA>128 THEN HA=HA-16Ø
```

Der erste Ton eines Taktes wird durch Anhebung der Lautstärke etwas betont.

```
274Ø   SV=12: IF L=1 THEN SV=15
275Ø   IF HA>91 THEN SD=D/16: SP=HA-32-5: GOSUB 263Ø: GOTO 284Ø
276Ø   IF MID$(T$(T),L+1,1)="*" THEN 281Ø
277Ø   SP=HA-5:SD=D/8:GOSUB263Ø:GOTO284Ø
278Ø   REM
279Ø   REM ** Triller **
28ØØ   REM
281Ø   SD=D/32: SP=HA-5
282Ø     FOR H1=1 TO 8: SP=HA+HA-9-SP: GOSUB 263Ø: NEXT H1
283Ø   L=L+1
284Ø NEXT L
285Ø RETURN
```

Die Programmausschriften sind wie der gesamte Menütext einem Datenfeld zugeordnet. Dadurch lassen sich Unterprogramme mehrfach nutzen.

```
29ØØ REM
291Ø REM ** Bildschirmsteuerung **
292Ø REM
293Ø READ SR$: W=W+1: W$(W)=SR$
294Ø HO= (HT-LEN(SR$))/2-3: GOSUB 11Ø
295Ø RETURN
3ØØØ REM
3Ø1Ø REM ** Zeile loeschen **
3Ø2Ø REM
3Ø3Ø VE=VT-2: SR$=W$(1Ø): HO=HT-8: GOSUB 11Ø: PRINT SR$:
3Ø4Ø RETURN
31ØØ REM
311Ø REM ** Textzeile **
312Ø REM
313Ø GOSUB 3Ø3Ø: VE=22: GOSUB 294Ø: PRINT SR$;: SR$=W$(X)
314Ø GOSUB 294Ø: PRINT SR$;: GOSUB 21Ø
315Ø RETURN
32ØØ REM
321Ø REM ** Zeile in inverse **
322Ø REM
323Ø VE=22: SR$=W$(1Ø): GOSUB 294Ø: PRINT SR$;: SR$=W$(X)
324Ø GOSUB 294Ø: GOSUB 15Ø
325Ø RETURN
33ØØ REM
331Ø REM ** Zeile printen **
332Ø REM
333Ø VE=VT-2: HO=HT-8: GOSUB 11Ø: PRINT RIGHT$(SR$,8);
334Ø RETURN
34ØØ REM
341Ø REM ** Zahl loeschen **
342Ø REM
343Ø VE=X*2+7: HO=HE: GOSUB 11Ø: PRINT W$(1Ø);: GOSUB 11Ø
344Ø INPUT A$: GOSUB 11Ø: PRINT W$(1Ø): GOSUB 11Ø: PRINT " ";A$;
345Ø RETURN
```

Für Tempo und Oktave sind Standardwerte vorgegeben. Zwecks Anpassung an den Computertyp, aber auch zum Experimentieren, sind diese Werte über das Menü austauschbar.

```
35ØØ REM
351Ø REM ** Tempo veraendern **
352Ø REM
353Ø GOSUB 343Ø: D=INT(VAL(A$)): IF (D<6) OR (D>99) THEN 353Ø
354Ø GOTO 153Ø
36ØØ REM
361Ø REM ** Oktave veraendern **
362Ø REM
363Ø GOSUB 343Ø: SO=INT(VAL(A$))
364Ø IF (SO<Ø) OR (SO>5) THEN 363Ø
365Ø GOTO 153Ø
```

Der interessierte Spieler wird versuchen, selbst Taktfolgen zusammenzustellen. Einer ganz ohne System, ein anderer vielleicht mit einem eigenen Zufallsgenerator (Würfel). Durch die weiter unten angeführten BASIC-Befehle läßt sich der Computer auch das Geheimnis seiner Takttabelle entlocken. Die Länge des Stückes bestimmt der Spieler durch Eingabe des Taktes "0" oder der Computer bei gefülltem Puffer.

```
37ØØ REM
371Ø REM ** Takteingabe **
372Ø REM
373Ø X=11: GOSUB 323Ø: X=3
374Ø GOSUB 343Ø: T=INT(VAL(A$))
375Ø IF (T<Ø) OR (T>119) THEN 374Ø
376Ø IF (T=Ø) OR (TZ>15Ø) THEN 388Ø
377Ø GOSUB 3Ø3Ø: SR$=T$(T): GOSUB 333Ø: GOSUB 271Ø: GOTO 374Ø
```

Wie sich jeweils 24 Takte unserer Walzer-Noten anhören, wenn sie im "ungeordneten Zufall" aneinandergefügt werden, erfahren Sie mit der Funktion RANDOM-SOUND. Bei dieser Gelegenheit ist auch mal ein Blick auf die ausgewählten Notenstrings gestattet.

```
38ØØ REM
381Ø REM ** Zufalls-Melodie **
382Ø REM
383Ø X=12: GOSUB 323Ø
384Ø FOR ZU=1 TO 24
385Ø   GOSUB26Ø: T=INT(RV*118)+1: GOSUB 3Ø3Ø
386Ø   SR$=T$(T): GOSUB 333Ø: GOSUB 271Ø
387Ø NEXT ZU
388Ø SV(TZ+1)=-99: GOTO 219Ø
```

Stehen Strings als Konstanten in den DATA-Zeilen, benötigen sie weniger Speicherplatz und erleichtern dem BASIC-Interpreter die Verwaltungsarbeit. In Zeile 1000 ist mit A = 1000 die Größe des Stringspeichers vorgegeben. Die KC-Rechner benötigen für das Programm "MOZART" diesen Speicher nicht. Sollten Sie bei Ihrem Computer Speicherplatzprobleme bekommen (DIM-Befehle), so versuchen Sie, einen passenden Wert mit A> =0 zu finden.

```
25ØØØ REM
25Ø1Ø REM  *** MENUEDATEN ***
25Ø2Ø REM
25Ø3Ø DATA "           ", "M O Z A R T", "-----------"
```

```
25Ø4Ø DATA "  KV 294d  ", "           ", "1 ... Tempo (6-99) :"
25Ø5Ø DATA "2 ... Oktave (Ø-5) :", "3 ... Takt(Ø-119)  :"
25Ø6Ø DATA "4 ... Random-Sound  ", "5 ... Mozart-Walzer "
25Ø7Ø DATA "6 ... Programmende  ", "-ich lerne Noten-"
25Ø8Ø DATA "      Sie waehlen  :", "-ich komponiere-"
25Ø9Ø DATA "-ich musiziere-", "                       "
251ØØ DATA "Ø = vorspielen", "So ein Zufall !", ">TASTE<"
```

BASICODE verleitet den Programmierer zu immer ausgefalleneren Ideen. Kürzer kann man Noten wohl kaum verschlüsseln:
Man ziehe sich Notenlinien und bezeichne den Ton "C" unter diesen Linien mit dem Buchstaben "A". Nun schreibe man das Alphabet nach oben, für jeden ganzen und jeden halben Ton je ein Buchstabe. Dadurch bekommt das nächste "C" den Buchstaben "M" usw.
So sind mehr als zwei Oktaven zu erreichen, was zumindest für dieses Spiel ausreichend ist. Abschließend wird noch festgelegt, daß kleine Buchstaben 1/16-, die anderen 1/8-Noten sein mögen. Folgt ein "*", so produziert das Programm einen Triller mit 8/32-Noten.

```
2511Ø REM
2512Ø REM *** Noten der Takte 1-119 ***
2513Ø REM
2514Ø DATA "ROT", "Jghlt", "TMQ", "TO*", "HMQ", "qmqtyt", "QOH"
2515Ø DATA "ljlmol", "qmljhg", "MMM", "MHE", "QtqM", "VSO"
2516Ø DATA "mhmqhm", "qmQT", "Tx{O", "mqtojs", "QMH", "rqoqrt"
2517Ø DATA "cgjosv", "QQQ", "rqroml", "sojvso", "lotoL", "qmHQ"
2518Ø DATA "qmolH", "JOS", "jqotsv", "txtoL", "MHQ", "THH"
2519Ø DATA "mlmqhm", "mlmqH", "lmoljh", "Trqom", "Jrojl", "mlmhea"
252ØØ DATA "Txtol", "TtoX", "Qmqty", "MqmH", "mhqmtq", "onorhl"
2521Ø DATA "QqrT", "TXO", "olHT", "TQM", "MqmT", "qmlhjg", "qmlmH"
2522Ø DATA "qtytqm", "ojOS", "SVS", "Txtot", "Svsos", "tx{xT"
2523Ø DATA "rqomlo", "yxytqm", "SSS", "txTO", "OJS", "olHT", "MQH"
2524Ø DATA "OotX", "totxto", "rqOT", "sv{vsv", "OtxO", "tstolh"
2525Ø DATA "SVO", "qoqtyt", "soJS", "mqmhE", "Jomlj", "QTY"
2526Ø DATA "ororlo", "ommllj", "ROL", "OOO", "orvrol", "ojosvs"
2527Ø DATA "qvtxsv", "qmtqyt", "[vsoj", "TxtO", "tstxO", "tqolH"
2528Ø DATA "mhmqtq", "LOT", "vtstO", "QolH", "Otolo", "jqomlh"
2529Ø DATA "SsoV", "SS*", "txtxO", "JjoS", "oqroml", "TolH"
253ØØ DATA "orjolo", "OSV", "qyxtvs", "roJL", "QQ*", "ojsovs"
2531Ø DATA "onosvs", "txtolh", "mhqmT", "Loljh", "qtomlj", "OHH"
2532Ø DATA "OHL", "oxtoL", "MmoQ", "qtotjs", "lmoqro", "rvJlo"
2533Ø DATA "qmloT", "vtxtot"
```

Die Zahlentafeln lassen sich besser lesen, wenn man die Felder WA(Z,WT) und WB(Z,WT) auf den Bildschirm oder Drucker bringt. Die Routine könnte lauten:

```
FOR Z=2 TO 12: FOR WT=1 TO 7: PRINT WA(Z,WT);
NEXT WT: PRINT: NEXT Z
```

Bedingung ist, daß die Felder eingelesen sind.

```
2534Ø REM
2535Ø REM  *** Zahlentafeln ***
2536Ø REM
2537Ø DATA18,55,18,85,98,21,33,7,55,18,86,18,9,76,5,81,5,31
```

```
25380 DATA89,88,50,46,101,5,38,5,93,80,47,53,47,96,109,12
25390 DATA11,91,105,47,45,47,77,100,32,67,14,119,78,6,63,26
25400 DATA79,32,2,32,110,43,58,20,58,99,69,51,37,87,61,58
25410 DATA69,58,28,1,71,95,71,56,22,104,114,24,106,71,54,71
25420 DATA82,19,35,13,35,107,116,18,44,118,70,35,92,35,17,66
25430 DATA83,84,42,65,79,40,41,113,97,42,64,42,115,57,30,52
25440 DATA30,60,34,15,108,62,59,30,90,30,49,103,3,72,3,4
25450 DATA112,25,48,111,94,3,39,3,102,117,10,27,10,16,8,75
25460 DATA73,68,23,10,29,10,74,36
30000 REM
30010 REM ***************************
30020 REM *                         *
30030 REM * Dieses Programm wurde   *
30040 REM * fuer BASICODE - 3 auf   *
30050 REM * einem KC85/4 entwickelt.*
30060 REM * Den Algorithmus entnah- *
30070 REM * men wir dem KV 294d von *
30080 REM * Wolfgang-Amadeus Mozart,*
30090 REM * der uns die Verstuemme- *
30100 REM * lung seines Werkes ver- *
30110 REM * zeihen moege.           *
30120 REM * In die vorliegende Form *
30130 REM * brachten die Noten:     *
30140 REM *                         *
32000 REM * Uwe und Andreas Zierott *
32010 REM *      Pappelallee 1      *
32020 REM *       Lehnin 1804       *
32030 REM *           DDR           *
32040 REM *                         *
32050 REM *                21.09.89 *
32060 REM *                         *
32070 REM ***************************
32080 REM
32090 REM
```

7. Die Geschichte

7.1. Die Entwicklung von BASICODE

Angefangen hat es 1979. Verschiedene Heimcomputer wurden produziert und verkauft. Teleac (TELEvision ACademie) strahlte einen Computerkurs aus. Schon damals waren die Besitzer von Heimcomputern stark an Software interessiert. So begann Radio Hilversum als festen Bestandteil einer wöchentlichen Sendung, Computerprogramme auszustrahlen. Damals waren vier Computertypen besonders verbreitet: der Apple II, der Exidy Sorcerer, der Commodore PET und der Tandy TRS-80 Model-1. In jeder Sendung wurde ein von einem Zuhörer geschriebenes Programm ausgestrahlt, abwechselnd für den Apple, Exidy usw. Die Programme wurden über UKW, also nahezu störfrei, ausgestrahlt. Wer die Sendung für seinen Computer auf Kassette mitgeschnitten hatte, konnte nachher Freude am Programm haben. Zuhörer, die keine Ahnung davon hatten, was ein Computer ist und wozu dieses schreckliche Geräusch diente, glaubten zuweilen, ihr Empfänger sei defekt.

Aber zuweilen gab es auch von Computerhobbyisten Beschwerden. Wenn man zufällig die Sendung für seinen Computer verpaßt hatte, mußte man vier Wochen warten, bis der eigene Typ wieder an der Reihe war. Besitzer anderer Geräte, wie z. B. des DAI, der damals in Holland an Bedeutung gewann, konnten nichts mit den Sendungen anfangen. Wurde eine Sendung irgendwie gestört, so waren die Aufzeichnungen nutzlos. Alles oder nichts ist zwar durchaus binär, aber dennoch kaum zufriedenstellend. Manchmal genügte es, daß auch nur ein Bit verändert wurde, um das Programm unbrauchbar zu machen. Und dann nur einmal in vier Wochen oder noch seltener erfolgreich zu sein, das genügte vielen nicht.

Eine wichtige Folge dieser Sendungen war aber, daß einige aktive Hobbyisten miteinander in Kontakt kamen. Man sprach über die Empfangsprobleme und suchte nach Lösungen hierfür. Anfang 1981 kam die entscheidende Idee, und zwar von Klaas Robers, der damals als Diplomingenieur im Philips Forschungslabor arbeitete. Die Sendungen sollten so gestaltet werden, daß sie für jeden Computer und nicht nur für einen genutzt werden konnten. Weiter sollte erreicht werden, daß (geringe) Störungen keine wesentlichen Folgen nach sich zogen. Trotz der Störung sollte das Programm zumindest in den Computer einlesbar bleiben. Jeder Hobbyist hatte damals soviel Programmierwissen, daß er kleine Veränderungen, die durch die Störung bedingt waren, selbständig rückgängig machen konnte.

Am Donnerstag, den 14. Mai 1981 war es dann so weit. Die aktiven Teilnehmer der Gruppe kamen zusammen und die Idee wurde besprochen. Jeder kannte seinen eigenen Computer bis zum letzten Bit und Chip. Das war auch notwendig, denn anderenfalls war es nicht möglich, die notwendigen, nicht einfachen Maschinen-Programme zu schreiben. Oft gab es noch keinen Assembler. Alles mußte direkt hexadezimal eingegeben werden. Das gemeinsame Arbeiten führte aber zum Erfolg. Die Pläne von Klaas Robers wurden besprochen, kommentiert und weitergeführt. Kurz vorher war über den Rundfunk eine Testsendung mit einem kleinen Programm abgestrahlt worden. Jeder hatte diese Sendung empfangen und ausgewertet. Im allgemeinen war man so begeistert, daß beschlossen wurde, auf diesem Weg weiterzugehen. Die weiter entwickelten Pläne wurden schließlich von allen bestätigt und so war BASICODE geboren!

Es folgten für jeden viele Wochen spannungsvollen Programmierens. Zu dieser Zeit war BASICODE nur eine Festlegung dafür, wie man ein BASIC-Programm auf das Tonband aufzeichnen sollte. Unterschiede zwischen den verschiedenen BASIC-Dialekten waren damals kein Problem, denn, wie bereits gesagt, hatte jeder so viel Kenntnis, daß er leicht ein empfangenes Programm paßgerecht abändern konnte. Es war schon äußerst aufregend, daß man das Programm eines anderen Computertyps einlesen konnte. Abends und an den Wochenenden des Sommers 1981 wurde an diesem Prinzip gearbeitet.

Am 11. August 1981 wurde in einer Pressekonferenz BASICODE der Öffentlichkeit bekanntgegeben. Alle Aktiven waren wieder anwesend und hatten ihre Computer mitgebracht. Und wirklich, das Wunder geschah. Eine Kassette, von einem Computer geschrieben, wurde einem anderen überreicht und tatsächlich in diesen Rechner eingelesen. Jeder Computer, ob grün oder blau, ob Apple oder Commodore, jeder produzierte die gleichen Signale. Schließlich hatte man dafür gearbeitet. Aber auch, wenn man es jetzt wirklich sah, war es noch immer fast unglaublich. Die Sprachenbarriere zwischen den Rechnern war bezwungen!

Ende September 1981 fing man damit an, die ersten Vertaalprogramme (Bascoder) auszustrahlen und im Oktober war das erste BASICODE-Programm dran. Bald gab es für mehr als zehn Computertypen einen Bascoder. In einer neuen Zusammenarbeit zwischen dem BASICODE-Team, dem Rundfunk und einem nationalen Computerklub wurde ein Buch geschrieben und dazu eine Kassette herausgegeben, die alle Bascoder enthielt. Fast ein Jahr nach der ersten Pressekonferenz konnte man dieses Ensemble kaufen. Innerhalb von wenigen Monaten war die erste Auflage von 2000 Stück ausverkauft.

Empfangsprobleme gab es jetzt kaum mehr und die gelegentlichen kleinen Störungen konnten einfach beseitigt werden. Der Heimcomputer wurde schnell populär und viele Menschen kauften sich so ein Gerät. Deshalb gab es auch immer mehr Besitzer, für die es zu schwierig war, die empfangenen Programme für den eigenen Computer zu modifizieren. So entstand die Frage, ob es vielleicht eine Möglichkeit gibt, BASICODE dahingehend zu erweitern, daß die ausgestrahlten Programme direkt, also ohne nachträgliche Bearbeitung, auf allen Rechnern ablauffähig sind?

Am 30. September 1982 kam daher die inzwischen beachtlich angewachsene und teilweise geänderte BASICODE-Mannschaft wieder zusammen. Jochem Herrmann, damals Student an der Technischen Universität Eindhoven, und Klaas Robers hatten Pläne dafür ausgearbeitet, wie eine Programmstruktur mit festgelegten Subroutinen aussehen könnte. Hierfür war es günstig, daß man während des ersten Jahres von BASICODE schon ziemlich genau die Probleme kennengelernt hatte. In jedem Programm benötigt man die Möglichkeit, den Bildschirm zu löschen und Zahlen auf dem Bildschirm zu schreiben. Leider benutzten aber die unterschiedlichen Rechner dazu abweichende Befehle. Mit festgelegten Subroutinen könnten diese Probleme beseitigt werden. Die ganze Mannschaft überlegte mit, welche Subroutinen notwendig wären. Weiter wurde beschlossen, Richtlinien über die Verwendung von Zeilennummern festzulegen.

Die Unterschiede zwischen den einzelnen Computertypen wurden genau analysiert. Es gab eine Reihe von BASIC-Befehlen, die bei allen Rechnern exakt übereinstimmten, und andere wenige, die völlig verschieden wirkten beziehungsweise auf einigen Rechnern nicht einmal vorhanden sind. Jochem Herrmann formulierte ein genaues Protokoll aller Subroutinen und aller zugelassenen BASIC-Befehle. Das Protokoll und ein Testprogramm wurden unter den Aktiven verbreitet. Das Testprogramm prüfte die Wirkung aller Subroutinen. BASICODE-2 war geboren.

Ein BASICODE-2-Buch mit Kassette wurde vorbereitet und erschien im Juli 1983. Das Buch enthielt das genaue Protokoll und die Gebrauchsanweisungen für die Bascoder. Das Protokoll wurde dann auch als BASICODE-Programm ausgestrahlt und später die neuen Bascoder. Die Hobbyisten waren begeistert. Zum ersten Mal konnte man Programme machen, die ohne weitere Bearbeitung auf verschiedenen Computertypen sofort liefen. BASICODE-2 hatte zwar seine Beschränkungen, aber die Vorteile überwogen doch erheblich.

In Computerzeitschriften, und nicht nur in den Niederlanden, sondern auch in der BRD und in England, erschienen Listings von BASICODE-2-Programmen und sogar komplette Bücher. Mit der Weiterentwicklung der Computertechnik wurden die Einschränkungen von BASICODE-2 deutlicher. Man verlangte allgemein mehr.

Am 11. Mai 1985 kam das BASICODE-Team wieder zusammen. BASICODE sollte erweitert werden zu BASICODE-3. Jacob Haubrich, Mathematiklehrer in Eindhoven, und Jochem Herrmann hatten umfassende Vorschläge: Musik, Graphik, inverse Textausgaben, Dateien, viele kleinere Erweiterungen. Alles sollte möglich werden. Es würde gewiß nicht einfach sein, aber man hatte im Laufe der Zeit ja auch viel gelernt.

Wieder ging jeder nach Hause mit einem gemeinsamen Ziel: BASICODE zu verbessern. Viele Hunderte Programmierstunden brachten völlig neue und immer bessere und schnellere Bascoder, einige davon ganz in Maschinensprache geschrieben. Das Resultat war 1986 die erste Auflage des BASICODE-3-Buches. Die Reaktionen waren Begeisterung. Viele Computerbesitzer fingen an, in BASICODE-3 zu programmieren. Insbesondere die graphischen Erweiterungen wurden gern genutzt. Programme, die nur Texte darstellten, fingen an zu langweilen. Auch jetzt, 1989, gewinnt BASICODE-3 noch täglich mehr Freunde, und das nicht nur in den Niederlanden!

7.2. Computersendungen im Rundfunk der DDR

Den größten Zulauf der im vorigen Abschnitt genannten BASICODE-Freunde haben wir bestimmt hier in der DDR zu verzeichnen. Seit Sommer 1989 werden es schätzungsweise mehrere zehntausend sein. Mit den Computersendungen im Rundfunk und zahlreichen anderen Aktivitäten haben wir wohl einiges dazu beigetragen. Dabei fing alles recht einfach an:

Es war ein Frühlingstag im Jahre 1986, da trafen sich Schüler der mathematisch-naturwissenschaftlich orientierten Hertz-Oberschule Berlin und diskutierten über die Problematik: Was ist Künstliche Intelligenz? Durch die führende Hand des Schulfunkredakteurs Steffen Malyszczyk von Radio DDR wurde daraus eine Sendung der "Jugendurania", natürlich mit Unterstützung eines kompetenten Gesprächspartners: Prof. Dr. Horst Völz vom Zentralinstitut für Kybernetik und Informationsprozesse der AdW der DDR.

Nachher sprach man noch in kleiner Runde, bei der auch Dr. Ursula Findeisen vom Zentralvorstand der Urania dabei war, über Möglichkeiten des Rundfunks, mit Sendungen insbesondere junge Hörer für Fragen der Informatik zu begeistern und Interessenten gewisse Hilfe zu geben - aber wie?

Ausländische Rundfunkstationen hatten schon das eine oder andere probiert, doch was könnten wir in der DDR tun? Es war ein Vorschlag von Prof. Völz, es am Anfang mit einem Programmierkurs BASIC zu versuchen, und er bot bereitwillig seine Unterstützung an. Da war sie, die Idee!

Doch bis zu ihrer Realisierung mußten noch viele Fragen überdacht und geklärt werden. Vor allem, besteht bei jugendlichen Hörern ein Bedürfnis für Rundfunksendungen zum Thema Computer/Informatik und gibt es ein gesellschaftliches Interesse daran? Aus heutiger Sicht erscheint diese Frage geradezu trivial, doch im Sommer 1986?

Es ist klar, daß eine solche Entscheidung nicht nur vom Wollen eines einzelnen engagierten Redakteurs abhängt, der sich in der Person des Schulfunkredakteurs für Naturwissenschaft, Dr. Joachim Baumann, schnell fand. Neuartige Projekte müssen geplant und verteidigt werden und Risikobereitschaft wurde nicht nur auf "unterer" Leitungsebene verlangt. Nicht zuletzt in dem Augenblick, als Autor und Redakteur der Meinung waren, daß man für die effektive Durchführung eines Programmierkurses auch Computerprogramme über den Rundfunk abstrahlen müßte. Ganz abgesehen von technischen Problemen, die dazu geklärt werden mußten, trat da natürlich die Frage auf, wie diese "quietschenden, piepsenden und sonstwie für das normale Ohr ungewohnten" Geräusche von den Hörern angenommen werden? - oder ob wir sie damit vertreiben?!

Eine Frage ganz anderer Natur: Wie gestaltet man einen Programmierkurs hörenswert? Wir standen auch hier vor absolutem Neuland.

Für eine Testsendung wurde das Wissenschaftsmagazin des Schulfunks am 16.10.1986 genutzt. Neben einem Interview mit Prof. Völz zum Thema Jugend und Computer gab es einen Beitrag "Grundregeln zum effektiven Programmieren in BASIC". Am Schluß wurden drei verschiedene BASIC-Programme zur Berechnung von Primzahlen ausgestrahlt - eine Premiere im Rundfunk der DDR. Ein Manuskript der Sendung konnte angefordert werden.

Und da war sie , die Bestätigung! Eine enorme Hörerresonanz (das Thema der Sendung wurde nicht vorher in der Presse veröffentlicht), die das Interesse an solchen Sendungen bestätigte, ja diese geradezu forderte. Kein Wunder, denn Computerliteratur war seiner Zeit noch rar, das Interesse an der neuen Technik durch die ständig wachsende Rolle der Computer in Wirtschaft, Beruf und Freizeit vorhanden und die Qualifizierungsmöglichkeiten (Berufsausbildung, Volkshochschule, Computerclubs...) zu gering. Auch die Wahl der Programmiersprache BASIC schien richtig. Ist doch BASIC auf Grund der Syntax und Struktur einfach zu erlernen und als Sprache in fast allen Computern, wenn auch mit gering abweichenden Dialekten, implementiert.

Was ergab der Test zur Einlesbarkeit der ausgestrahlten BASIC-Programme? Die technische Erprobung erfolgte mit Unterstützung der Studiotechnik Rundfunk. Hierfür wurden die Programme vom Computer (KC85/3) auf Studioband (38 cm/s, mono, zweikanalig) aufgezeichnet und bei der Ausstrahlung von allen Funkhäusern und Sendern auf Kassette mitgeschnitten. Die Auswertung (Einlesbarkeit der Programme) war zufriedenstellend. Wir waren uns unserer Sache sicher, zu sicher, wie sich bald herausstellte.

Die Sendung vom 16.10.86 beseitigte letzte Bedenken, das Projekt "Programmierkurs" wurde konsequent angegangen. Bis Jahresende entstand eine Konzeption (Prof. Völz) für 20 Folgen "BASIC 1 x 1 des Programmierens - vom Taschenrechner zur Computergrafik". Eine Erkennungsmelodie aus den Noten B-A-S-C (leider gibt es kein I) wurde speziell komponiert, natürlich auf einem Musikcomputer! In jeder Sendung sollten ein oder mehrere BASIC-Befehle erläutert (Syntax) und ihre Wirkungsweise anhand kleiner Programme demonstriert werden. Dabei wurde auf den BASIC-Dialekt der KC85/x zurückgegriffen. Die Programme der Sendungen sollten vollständig ausgestrahlt werden, so daß der Hörer sie auf Tonband oder Kassette aufnehmen und nutzen könnte.

Am 7.1.1987 war es dann soweit. Innerhalb des Schulfunkprogramms hatte die erste Folge von "BASIC - 1x1 des Programmierens" Premiere. Ein schriftliches Begleitmaterial zum Kurs konnte den Hörern kostenlos angeboten werden. Eine bis dahin nicht gekannte Flut von Briefen und Materialwünschen traf in der Redaktion ein. Wir hatten es geschafft! BASIC war von Anfang an ein "Renner" bei den computerinteressierten Hörern, und das sind vor allem Jugendliche, Berufsschüler, Lehrer, Studenten, Facharbeiter der EDV ...

Der Volksmund besagt, daß die ersten Pflaumen madig sind. Nun, auch wir hatten Probleme. Viele Hörer schrieben der Redaktion, daß die Programme, die am Radio aufgezeichnet wurden, nicht in den Rechner einlesbar sind. Im Prinzip verwundert das nicht, denn Technik ist empfindlich und auf analoge Audiosignale ausgerichtet. Digitales war nie erprobt und bisher kein Ziel gewesen. Außerdem bedenke man, welchen Weg das Computersignal geht, um vom Studio (in Berlin) zum Hörer (z. B. Stralsund oder Sonneberg) zu gelangen. Überall treten Störungen verschiedenster Art auf, sowohl auf der Sendestrecke wie auch beim Empfang. Hundertprozentige Sicherheit kann es da nicht geben. Außerdem stellten wir fest, daß z. B. der KC85/3 die Signale besser annimmt als der KC85/1 oder Z1013 (die Tests nach der Sendung vom 16.10.86 wurden aber nur mit einem KC85/3 durchgeführt).

Nach vier Sendungen "BASIC - 1x1" wurde daher die Aufzeichnungstechnik umgestellt (stereo, einkanalig). Das Ergebnis war eine wesentlich verbesserte Einlesbarkeit der Programme, doch völlig störungsfrei sind sie trotzdem nicht.

Wie auch immer, mit den 20 Folgen "BASIC - 1x1" sowie den weiteren Sendeprojekten auf dem Gebiet der Informatik hat der Schulfunk von Radio DDR eine interessierte Stammhörerschaft gewonnen, die die Sendungen nicht nur dazu nutzt, die eigene Software-Bibliothek aufzufüllen, sondern durch aktive Teilnahme tiefer in das interessante Gebiet der Computertechnik einzudringen. Der Rundfunk kann hierbei Hilfestellung leisten. Seine Möglichkeiten sind verschiedenartig, aber auch nicht unbegrenzt. Uns wurde auch bekannt, daß nach diesem Lehrgang eine Vielzahl von Computer-Kabinetten ihre Arbeit gestaltete und häufig die Ausbildung der "Neuen" aufbaute.

Hörernähe - sie ist von besonderer Bedeutung. Das heißt, daß Wege gefunden werden müssen, den Hörer aktiv in das Programm einzubeziehen. Dabei ist die Beantwortung von Hörerfragen (schriftlich oder in den Sendungen) genauso wichtig wie der Kontakt zu Computerclubs, das Auftreten in öffentlichen Veranstaltungen usw. Als äußerst nützlich hat sich der Softwaretausch "per Radio" erwiesen. Computerfreunde schicken Programme an den Rundfunk, wovon die besten zur Ausstrahlung gelangen. Somit muß nicht jedes Fahrrad mehrfach erfunden werden, und es bleibt Zeit für eigenes.

Wichtig und notwendig ist ebenfalls die Zusammenarbeit mit anderen Medien. So konnte z. B. mit dem URANIA-Verlag die Herausgabe eines umfangreichen Sonderheftes "URANIA-Extra" noch im Sommer 1987 realisiert werden, in dem der Rundfunkkurs "BASIC - 1x1 des Programmierens" in wesentlich größerem Umfang als im Begleitmaterial zu den Sendungen abgedruckt wurde. Das Heft wurde in 200 000 Exemplaren verkauft.

In Zusammenarbeit mit dem VEB Deutsche Schallplatten wurde der Kurs auf Kassette produziert (einschließlich der Computerprogramme, die z. T. für Atari, C64 und ZX-Spectrum angepaßt wurden) und als Präsentbox in weit über 10000 Exemplaren verkauft. Insbesondere Betriebsberufsschulen und Oberschulen der Volksbildung zeigten großes Interesse, da diese Einrichtungen nicht immer die Gelegenheit haben, "aktuell" am Radio mitzuschneiden und sie für die eigene Ausbildung einzusetzen.

Die Entwicklung der Informatiksendungen im Rundfunk darzustellen heißt, einer gewissen Chronologie zu folgen. Das Interesse war geweckt, die Notwendigkeit bestätigt. Es hieß also: weitermachen! Doch nicht nur zeitlich, sondern auch inhaltlich. Deshalb wurde folgender Weg gewählt:

Innerhalb des Schulfunkprogramms sollte es 1988 wöchentlich etwas zur Informatik geben. Aufbauend auf den Kurs "BASIC - 1x1" wurde in bewährter Zusammenarbeit mit Prof. Völz ein zehn Folgen umfassender Lehrkurs "BASIC für Fortgeschrittene" gestaltet. In jeder Folge wurden, inhaltlich abgeschlossen, ausgewählte Themen behandelt. So z. B. Sortieralgorithmen, Kombinatorik, Einbinden von Maschinencode in BASIC, lernfähige Programme usw. "BASIC für Fortgeschrittene" gab es einmal monatlich. Auch für diesen Kurs, wie übrigens für alle weiteren, konnten die Hörer kostenlos ein schriftliches Begleitmaterial erhalten und so ein besseres Verständnis der Sendungen erreicht werden.

Zweimal im Monat wurde "BASIC - 1x1" wiederholt, um auch "Neueinsteigern" die Möglichkeit zu geben, sich das Grundwissen anzueignen.

In der 4. Woche gab es dann den "Software-Service". Hier wurden speziell Programme ausgestrahlt, die Hörer an die Redaktion schickten. Darunter recht professionelle Software, wie z. B. Grafik- oder Kalkulations-, Datei-, Dienst- und Spielprogramme. Insbesondere der "Software-Service" hilft, das Softwareloch etwas zu stopfen. Denn die von den Computerherstellern zum Verkauf für die Kunden bereitgestellte Software ist quantitativ viel zu gering.

Dreimal (in der 5. Monatswoche) gab es 1988 ein BASIC-Extra im Schulfunk. Diese Sendungen wurden vorwiegend zur Beantwortung von Hörerfragen genutzt.

Die wesentlichen Bestandteile der Sendungen wurden im Computerclub von Jugendradio DT 64 wiederholt. Das begrüßen viele Hörer, die nicht immer die reguläre Schulfunkzeit (17.00 Uhr) verfolgen können. Ein weiterer Vorteil: die Qualität der Übertragung von Computersignalen ist bei den beiden Sendern unterschiedlich, so daß man wählen kann, welcher Sender für den eigenen Standort die Programme besser überträgt.

Unter dem Motto: "Jedes Jahr etwas neues" stellten wir Überlegungen an, was 1989 folgen sollte. Wir wendeten uns auch an die Hörer. Eine schriftliche Umfrage brachte das Ergebnis:

Fast 90% wünschten sich einen Programmierkurs in Maschinencode, Pascal oder Forth. Außerdem wurde Interesse an Informationen zu neuer Computertechnik und -literatur bekundet, Erfahrungen aus Computerclubs sowie Hard- und Software-Tips sollten in den Sendungen vermittelt werden und und und.

Es reifte logisch die Idee, ein Computermagazin zu schaffen. Hier kann man all das reinpacken. Bewährtes (Kurse, Software-Service) wird beibehalten, Neues (Informationen, Tips, Hörerecke) kommt hinzu. Und somit erhielt (nach reichlich langem Nachdenken) das Kind einen Namen: **"REM - das Computermagazin".**

REM ist ja eine bekannte BASIC-Anweisung. Sie dient dazu, dem Nutzer Informationen, Erläuterungen und Hilfen zu vermitteln - das gleiche Ziel hat "REM".

REM ist seit Beginn an, also seit Januar 1989, eine durchmoderierte halbstündige Sendung. Neben den zahlreichen Computerprogrammen für die unterschiedlichsten Rechner (KC85/2-4, KC87, Z1013, AC1), vielen Tips, Informationen und Gesprächen mit Computerfreunden gab es ab März den Kurs "Einführung in Maschinencode" für den Mikroprozessor U880/Z80, sieben Folgen in bewährter Form (Prof. Völz).

Was seit Beginn der Sendungen, eigentlich seit dem 16.10.1986, bleibt, ist das leidige Problem der Programmübertragung für die verschiedenen Computertypen, bedingt durch die Verschiedenheit der Befehle und noch stärker durch die Unterschiede in der Signalcodierung.

An dieser Stelle soll ausnahmsweise ein Stammhörer der Computersendungen namentlich genannt werden. Es ist Friedrich Dormeier aus Berlin (West). Er machte den Rundfunk der DDR darauf aufmerksam (Brief vom 8.10.87), daß es in den Niederlanden eine Lösung gibt, viele Computer unter einen Hut zu bekommen und daß schon jahrelange Erfahrungen mit der Ausstrahlung von Programmen in **BASICODE** vorliegen.

Noch vor Jahresende '87 wurde der Kontakt zur Stiftung BASICODE aufgenommen. Seitdem besteht zwischen dem Rundfunk der DDR und der Stiftung BASICODE ein kontinuierlicher Erfahrungsaustausch, wobei dieser insbesondere am Anfang nur in eine Richtung erfolgte. Im Sommer 1988 begannen Uwe und Andreas Zierott als erste, für DDR-Rechner einen Bascoder zu entwickeln. Von ersten Erfolgen konnten sich Klaas Robers und Jacob Haubrich bei einem Aufenthalt in der DDR im Oktober '88 überzeugen. Es begann nun die intensive Phase, die Bascoder auch für andere Rechner zu schreiben, wobei Andreas Graf (Atari), Frank Heyder (AC1) und Martin Duchrow (Z1013) erst später dazukamen (Martin hat sogar erst Ende April '89 begonnen!).

Seit September 1989 wird nun BASICODE im Rundfunk der DDR genutzt. Dazu wurde innerhalb von REM ein Programmierkurs "Einführung in BASICODE" durchgeführt. Es versteht sich, daß Prof. Völz wieder seine Unterstützung zusagte.

Unterstützung hierfür erhielten wir aber auch durch die Stiftung BASICODE, insbesondere von Jacob Haubrich.

Die Begeisterung ist bei den Computerfreunden in der DDR bestimmt genauso groß wie seinerzeit in den Niederlanden. Bis Ende Oktober 1989 wurden 10 000 Exemplare des Begleitmaterials zum Kurs "Einführung in BASICODE" verschickt. Unzählige Briefe und Telefonate bewiesen, daß die Computerfreunde von Anfang an begriffen, welch riesige Möglichkeiten eines Datentransfers via BASICODE bestehen.

Noch etwas war neu - die Produktion von Computerprogrammen auf einer Schallplatte. Als erstes Projekt hierfür wurden die Bascoder für einige in- und ausländische Rechner auf eine Testplatte gepreßt. Testpersonen waren die REM-Hörer selbst, und ihre Ergebnisse konnten mit genutzt werden, um die Platte zu diesem Buch in besserer Qualität herzustellen.

Mit BASICODE, davon sind alle Beteiligten überzeugt, kann ein qualitativer und quantitativer Sprung auch in den Rundfunk-Computer-Sendungen erreicht werden. Wir bitten alle Leser und Hörer, uns mit Ideen und interessanten BASICODE-Programmen zu unterstützen.

8. Die Rechte der BASICODE-Autoren

Die Geschichte von BASICODE ist im vorigen Kapitel dieses Buches dargestellt. Aus ihr wird deutlich, daß der Ursprung dieses Programmsystems in den Niederlanden liegt. Die Autoren haben sich dort in einer Stiftung zusammengeschlossen und zur Förderung des BASICODE-Gedankens auf persönliche materielle Vergütungen verzichtet. Einnahmen fließen der Stiftung zu, die daraus ihre Unkosten finanziert, wie z. B. Materialverbrauch, Reisekosten u. ä. Der Verzicht der Autoren auf eine Vergütung ist aber nicht gleichbedeutend damit, daß sie keine Rechte an den von ihnen mit hohem Arbeitsaufwand geschaffenen Bascodern und Programmen haben. In den Niederlanden und in vielen anderen Ländern besteht Urheberrechtsschutz für Computersoftware.

Was bedeutet dies nun? Historisch gesehen entstand in Europa im 19. Jahrhundert das Bedürfnis und die Notwendigkeit für Komponisten und Schriftsteller, sich mit juristischen Mitteln gegen eine unerlaubte Vervielfältigung ihrer Werke zu schützen. Unterstützt wurden sie in ihrem Streben von den Verlegern, die ebenfalls ein jedoch überwiegend materielles Interesse an solch einer Schutzform hatten. Ihnen gelang es schon weit früher, ihre Interessen über Privilegien teilweise zu sichern. In der bürgerlichen Rechtslehre entstand die Theorie vom geistigen Eigentum. Da ein Schutz der Urheberrechte nur national wenig wirksam war, wurde schon 1886 in Bern ein internationales mehrseitiges staatliches Abkommen zum Schutze der Rechte der Urheber an ihren Werken der Literatur, Kunst und Wissenschaft (Berner Übereinkunft) geschlossen. Im Jahre 1952 wurde auf Initiative der UNESCO das Welturheberrechtsabkommen (WUA) ins Leben gerufen.

Sowohl die Berner Übereinkunft als auch das Welturheberrechtsabkommen wurden zwischenzeitlich mehrfach verändert, insbesondere natürlich die Berner Übereinkunft, die deshalb heute auch als Revidierte Berner Übereinkunft (RBÜ) bezeichnet wird. Es wurden in den verschiedenen Revisionen die Rechte der Urheber erweitert, aber auch neue Werkformen mit aufgenommen, die durch technische Entwicklungen möglich wurden, so z. B. die Fotografie, der Film, der Tonfilm usw. Wer sich näher über diese Fragen informieren möchte, kann zu Meyers Taschenlexikon "Urheberrecht" greifen, das seit 1975 im VEB Bibliographisches Institut Leipzig in mehreren Auflagen erschienen ist.

Eine neue im Entstehen begriffene Werkform, die dem Schutz des Urheberrechts zugänglich sein soll, ist die Computersoftware. Da die DDR Mitglied der beiden Urheberrechtskonventionen (RBÜ und WUA) ist, können sich die niederländischen Programmautoren bei Verletzung ihrer Rechte auf sie berufen [21].

Die Autoren dieses Buches haben die Genehmigung erhalten, BASICODE kostenlos anzuwenden und neue Übersetzungsprogramme für die in der DDR vorhandenen Computer zu entwickeln. Auf dieser Basis wird die Initiative der Computerfreunde in unserem Lande gefördert, eigene Anwenderprogramme unter Beachtung der BASICODE-Regeln zu schreiben. Hierbei handelt es sich um eine sinnvolle Freizeitbeschäftigung und eine persönliche Weiterbildung. Sie darf jedoch nicht einem kommerziellen Zweck dienen. Daher erhält auch jeder Käufer dieses Buches die Übersetzungs- (Bascoder) und Beispielprogramme sowohl der niederländischen als auch der DDR-Autoren auf Schallplatte kostenlos. Der Preis des Buches ist so kalkuliert, daß für die Datenplatte nur die Herstellungskosten eingehen. Es ist deshalb auch erlaubt und gewollt, daß im häuslichen Bereich Teile der Programme kopiert und an Freunde kostenlos weitergegeben werden, damit auch sie an BASICODE teilnehmen können. Wer allerdings meint, daraus ein "Geschäft" machen zu wollen, verstößt sowohl gegen moralische Grundsätze als auch gegen eine Reihe von Bestimmungen, vor allem gegen unser Zivilrecht. Das gleiche gilt bei der Verletzung des Rechts auf Namensnennung. Soweit ein Programmautor genannt ist, darf seine Autorenschaft bei der Kopie nicht entfernt

werden. Es ist zwar erlaubt, Programme weiterzuentwickeln, aber es ist nicht erlaubt, so eine weiterentwickelte Version irgendwie weiterzugeben oder zu verbreiten ohne Einverständnis des ersten Autors. Es ist und bleibt ja sein Programm und nur er hat das Recht, sein Programm weiterzuentwickeln und eine neue Version zu verbreiten. Falls jemand anderes eine Weiterentwicklung hat, kann er diese nur als Vorschlag dem Originalautor vorlegen. Dabei ist aber neben dem eigenen Namen auch der Name des ursprünglichen Programmautors zu nennen. Die Schlußkennzeichnung der BASICODE-Programme bietet hierfür bis zur Nennung der geistigen Anteile gute Voraussetzungen. Wird z. B. ein Programm eines holländischen Autors nur in die deutsche Sprache übertragen, so könnte dort nach dem holländischen Autor etwa stehen: Übersetzung ins Deutsche (Name des Autors), gefolgt vom Datum. Übersetzungen ins Deutsche sind auch nur erlaubt, wenn der Originalautor damit einverstanden ist. Über die Stiftung BASICODE wird diese Genehmigung an Radio DDR weitergegeben, aber nur für bestimmte, festgelegte Zwecke: Rundfunk und Sammelkassette, und nur unter der Bedingung, daß daraus keine Geschäfte gemacht werden.

Es sollte Regel sein, daß der Autor des Programms und das Herstellungsdatum ausgewiesen werden. Damit soll seine Leistung in der Öffentlichkeit moralisch anerkannt werden. Andererseits verpflichtet ihn dies natürlich auch zur Qualitätsarbeit.

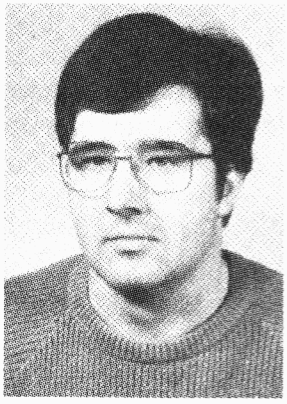
Alle diese Fakten führen zu dem Ergebnis, daß Programme in BASICODE, auch Anwenderprogramme, nicht verkauft werden dürfen. BASICODE setzt die Beachtung der Regeln voraus und vor allem das Vorhandensein geeigneter Übersetzungsprogramme. Ohne diese sind die Anwenderprogramme wertlos. Die geistigen Grundlagen von BASICODE und die Bascoder wurden von ihren Autoren der Öffentlichkeit kostenlos zur Verfügung gestellt, aber keineswegs zu dem Zweck, daß andere daraus Geschäfte entwickeln. Wer mit Programmieren Geld verdienen möchte, kann dies auf der Basis der vorhandenen gesetzlichen Bestimmungen, aber nicht unter Verwendung von BASICODE. Wird diese Regel verletzt, werden die Autoren dieses Buches auch mit juristischen Mitteln dagegen vorgehen.

Der Autor hofft, einige juristische und moralische Fragen im Zusammenhang mit BASICODE erläutert zu haben. Sie sind aber nicht mechanisch übertragbar auf andere Gebiete der Softwareautorenschaft.

9. Vorstellung der Autoren

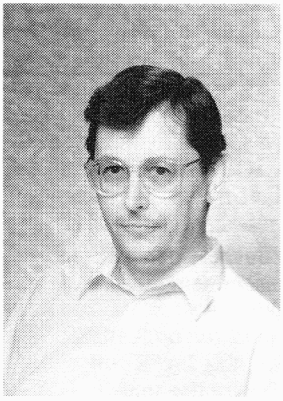
In Anlehnung an [1] sind hier einige Fakten zu den Autoren dieses Buches, den Autoren der Bascoder und anderen für dieses Buch wichtiger Persönlichkeiten zusammengestellt. Weitere befinden sich u. a. auf der Impressumseite.

9.1. Holländische Autoren

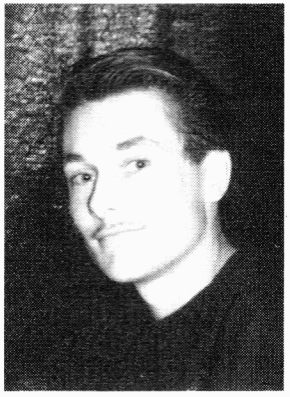


Jan Bredenbeek (1964)
Beruflich als Elektroniker tätig. Entwickelte die Bascoder für den Sinclair ZX-Spectrum und Sinclair QL.

Melis van Deelen (1965)
Computerhobbyist, beschäftigt sich mit BASICODE auf diversen Commodore-Computern. Verbesserte den C-64 BASCODER für BASICODE-3 und schrieb auch die deutsche Version.

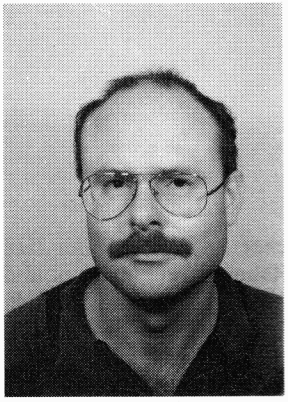


Jacob Haubrich (1942)
Mathematik- und Informatiklehrer in Eindhoven. Beschäftigt sich seit 1961 mit Computern. Entwickelte BASCODER für verschiedene Computer (PET, CBM 3000-, 4000- und 8000, VC-20, C-64) von BASICODE-1 bis BASICODE-3. Er ist Mitentwickler des BASICODE-2- und BASICODE-3-Protokolls. Bei Radio TROS bearbeitet er alle eingesandten Programme und ist wesentlich für alle Computersendungen verantwortlich. In der Stiftung BASICODE ist er für die Finanzen verantwortlich.

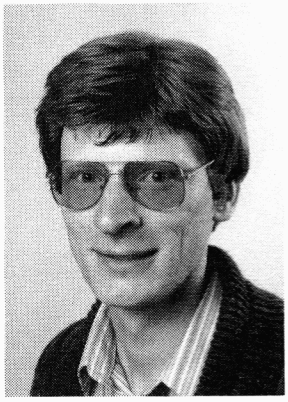


Robert Nico Mast (1970)
Beschäftigte sich bereits mit 9 Jahren mit LEGO. Zeichnet oft und viel. Schüler im Leistungskurs für Mathematik, Physik und Chemie, studiert jetzt an der Universität Amsterdam Bio-Chemie, möchte promovieren. Schrieb den Bascoder für den C plus 4.

Klaas Robers (1944)
Dipl.-Ing. bei Philips in Eindhoven. Studium Technische Universität Delft. Begeisterter Funktamateur. Erfinder des Systems BASICODE. Vorsitzender der Stiftung BASICODE in Holland.

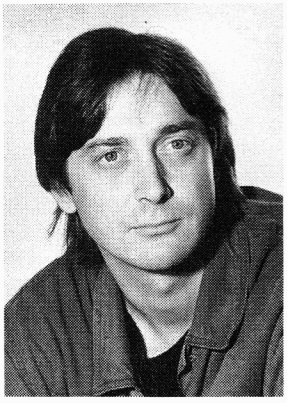


Sjef Simons (1953)
Arbeitet als Psychologe und beschäftigt sich seit 1981 mit Computern. Entwickelte die BASCODER für die Schneider-Amstrad CPC-Serie.

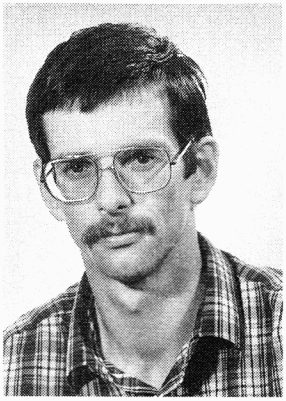


Hans Vosman (1961)
Diplom-Elektroniker Technische Universität Twente. Arbeitet mit an der Herstellung von Telekommunikationsgeräten. Hobbys: Elektrotechnik, Programmieren und Fotografie. Entwickelte zunächst den BASCODER für die Advance-B-Computer. Darauf aufbauend wurde von ihm die Voraussetzung geschaffen, um BASICODE-3 auf 16-Bit-PCs zu nutzen.

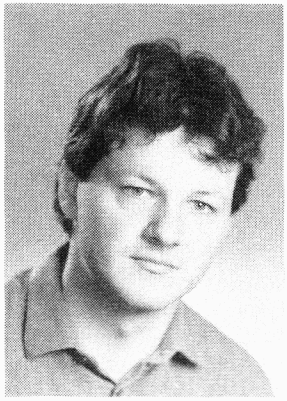
9.2. DDR-Autoren



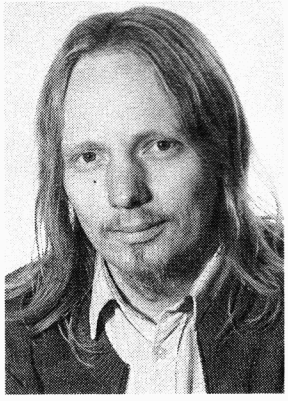
Dr. Joachim Baumann (1954)
Physikstudium (theoretische Festkörperphysik) in Odessa; Assistent an der TH Ilmenau; Promotion am Zentralinstitut für Physikalische Chemie der AdW der DDR. Seit 1984 Redakteur für Naturwissenschaft/Technik im Schulfunk von Radio DDR II, seit 1987 verantwortlich für die Informatiksendungen - REM-Moderator.



Andreas Böhme (1961)
Dipl.-Ing. und befristeter wissenschaftlicher Assistent an der IHS Berlin-Wartenberg. Seit 1987 nutzt er im Rahmen von Forschung und Lehre einen C 64. In dieser Zeit wurden von ihm eine Vielzahl anwenderspezifischer Basic-Programme für den C 64 geschrieben. Im Rahmen dieses Buches erprobte er den existierenden deutschen Bascoder und fertigte die vorliegende Beschreibung an.



Martin Duchrow (1954)
Dipl.-Ing.-Ök. Computerhobbyist seit 1986. Entwickelte den Bascoder für den Z 1013 und schrieb die Anleitung dafür.



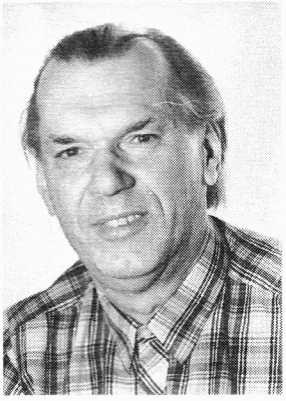
Andreas Graf (1958)
Diplomphysiker, wissenschaftlicher Mitarbeiter an der Akademie der Wissenschaften der DDR. Er beschäftigt sich mit Systemsoftwareentwicklung für spezielle Rechnersysteme. Er schrieb den Bascoder für die ATARI-Typen XL und XE, wobei auch ein spezieller BASIC-Interpreter entstand. Weiter fertigte er die hier vorliegende Beschreibung des Bascoders an.

Frank Heyder (1955)
Studium TH Ilmenau, Dipl.-Ing. für Informationstechnik. Gruppenleiter in der Hard- und Softwareentwicklung. Computerhobby seit 1981, Entwickler des Amateurcomputers AC 1, Mitglied der Clubleitung des Amateurcomputerclubs Berlin, Funkamateur mit besonderem Interesse für die technischen Betriebsarten RTTY, Packet Radio usw. Entwickelte mit B. Nickel den Bascoder für den AC 1.

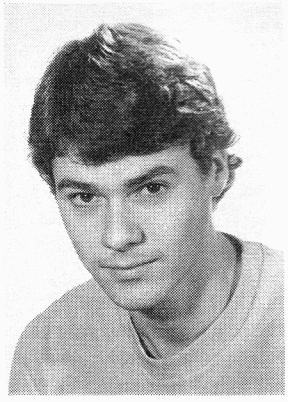
Marius van der Meer (1950)
Physikstudium Humboldt-Universität zu Berlin. Arbeitete mehrere Jahre auf dem Gebiet der amorphen Halbleiter, jetzt in der Sektion Elektronik zu modernen Rechnerarchitekturen (insbesondere Mehrprozessorsysteme und Neuronale Netze). Auch in der Informatik-Lehre tätig. Bearbeitete die deutsche Fassung der Bascoder für den ZX-Spectrum und mit zusätzlichen, wesentlichen Änderungen den für die CPC-Rechner.

Bodo Nickel (1951)
Ingenieurschule Berlin-Lichtenberg, Ing. für Nachrichtentechnik. Tätig als Ing. für Zuverlässigkeit. Computerhobby seit 1986, Eigenbau AC 1 (mit Disketten CP/M), besondere Neigung zur Erstellung von Tools für AC 1. Entwickelte mit F. Heyder den Bascoder für den AC 1.

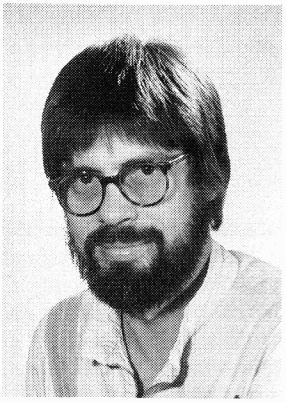
Peter Reichelt (1963)
Toningenieur beim Rundfunk der DDR. Arbeitet bei den Sendungen REM von Radio DDR 2 mit. Bearbeitet die deutsche Fassung für C plus 4.



Prof. Dr. Horst Völz (1930)
Rundfunkmechaniker, Physiker, Spezialist für Speichertechnik, Elektronik und Informationstheorie, arbeitet am Zentralinstitut für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der DDR. Sein Interesse gilt neben den technischen, theoretischen und Grundlagen-Problemen der Informatik betont auch Fragen der Ethik und Ästhetik. Umgang mit der Rechentechnik seit etwa 1975. Arbeitet mit BASICODE vorrangig auf einem EC1834 und einem IBM-kompatiblen PC.



Andreas Zierott (1969)
Facharbeiter für Datenverarbeitung, arbeitet im Rechenzentrum des Meteorologischen Dienstes der DDR. Umgang mit Computern seit 1986. Programmiert vorwiegend in Assembler, und erstellte schon frühzeitig Maschinenprogramme, die auf Routinen des BASIC-Interpreters zurückgriffen. Bekannt ist sein Programm Mozart für den KC 85/3. Eine BASICODE-Version in vereinfachter Form enthält Kapitel 6. Zusammen mit Uwe Zierott entwickelte er den Bascoder für die Kleinstrechner KC87, KC85/3, KC85/4 sowie die Vorgängertypen.



Uwe Zierott (1954)
Facharbeiter für Datenverarbeitung, arbeitet im Kundendienst einer KC-Werkstatt. Nebenberuflich als Dozent für Informatik an der Volkshochschule tätig. Umgang mit Computern seit 1985. Erstellt vorwiegend Systemprogramme und komplexe Lösungen für die Kleincomputer-Typen aus Mühlhausen. Zusammen mit Andreas Zierott entwickelte er den Bascoder für die Kleinstrechner KC 87, KC 85/3, KC 85/4 sowie die Vorgängertypen.

10. Hinweise zur Nutzung der Schallplatte

BASICODE ist nur dann leicht und zuverlässig anwendbar, wenn für alle Rechner die entsprechenden Bascoder verfügbar sind. Zum Teil sind diese Programme sehr umfangreich. Bisher wurden die Programme teilweise über den Rundfunk, über Kassetten und über eine Testschallplatte verbreitet. Die Schallplatte hat sich dabei aus mehrerlei Gründen als günstig herausgestellt. Sie wird daher auch als Beilage zu diesem Buch verwendet. Dies bedarf einer etwas ausführlicheren Begründung, und es sind Hinweise notwendig, wie mit der Schallplatte umzugehen ist.

Es gibt heute viele Methoden, Daten für Rechner zu speichern. Für eine Breitenanwendung von BASICODE war es notwendig, ein besonders preiswertes Verfahren zu finden. Sonst wäre dies Buch zu teuer geworden.

International hat sich am meisten die Diskette für die Verbreitung von Rechnerdaten und Programmen auch im Zusammenhang mit Büchern bewährt. Sie ist jedoch relativ teuer und verlangt ein aufwendiges Diskettenspeichergerät, was bei einer Vielzahl von Kleinstrechnern nicht vorhanden ist und nur von sehr guten Spezialisten mit hohem technischem Können angeschlossen werden kann. Eine weit verbreitete Variante sind Audiokassetten. Sie werden auch von der holländischen BASICODE-Stiftung im Zusammenhang mit dem Buch [1] eingesetzt. Sie besitzen jedoch für unser Anliegen zwei Nachteile: Sie sind wesentlich teurer in der Herstellung und lassen sich nur schwer Büchern und Broschüren als Anlage anfügen. Mit der kleinen Schallplatte (single) werden die folgenden Vorteile erreicht:

- Das Material und die Herstellung sind extrem preiswert.
- Die Wiedergabe kann mit jedem Plattenspieler erfolgen.
- Die single ist leicht an Bücher und Broschüren anzufügen.
- Guter Zugriff zu den durch Rillenabstand gekennzeichneten Teilprogrammen.
- Stabiles unveränderbares back-up-Medium.

Für die Wiedergabe der Information bestehen zwei Möglichkeiten:

- Mit einem kleinen Interface können die Signale direkt in den Rechner eingelesen werden. Hierbei sind meist ein Bandpaß von einigen hundert Hertz bis zu wenigen kHz und eine Verstärkung mit Pegelanpassung notwendig. Die Daten dürften für die Bascoder je nach dem Rechnertyp und dessen Kassetteninterface unterschiedlich sein. Für die BASICODE-Programme könnte sich eine einheitliche Variante bewähren.
- In den meisten Fällen ist es günstiger, die Information einmalig von der single über den Plattenspieler auf Kassette umzuzeichnen. Hierbei ist gleichzeitig eine Auswahl des für Sie erforderlichen Bascoders auf eine Kassette und der BASICODE-Programme auf eine zweite Kassette sinnvoll. Die Schallplatte wird dann gut als back-up-Medium geschützt aufbewahrt.

Nach den bisherigen Erfahrungen ist die zweite Variante in mehrfacher Hinsicht günstiger. Der Plattenspieler ist in den meisten Haushalten ohnehin mit einem Kassettenrecorder gekoppelt. Dadurch bereitet das Umspielen - eventuell auch bei Freunden und Bekannten - keine speziellen Probleme. Es sind jedoch einige Besonderheiten zu beachten. Die Schallplatte ist in reiner Seitenschrift, also in Mono aufgezeichnet. Bei Stereo-Geräten ist es vorteilhaft, wenn Sie die Monotaste betätigen oder einen Kanal abschalten.

11.Tabellen

Im folgenden sind einige wesentliche Daten für den Gebrauch von BASICODE zusammengestellt. Sie sollen so etwas wie ein kurz gefaßtes Kompendium darstellen. Die meisten Fakten sind im Haupteil des Buches ausführlicher dargestellt. Der Zugriff erfolgt am besten durch das Inhalts- und Sachwörterverzeichnis.

11.1. In BASICODE gültige BASIC-Befehle

ABS	AND	ASC	ATN	CHR$	COS
DATA	DIM	EXP	FOR	GOSUB	GOTO
IF	INPUT	INT	LEFT$	LEN	LET
LOG	MID$	NEXT	NOT	ON	OR
PRINT	READ	REM	RESTORE	RETURN	RIGHT$
SGN	SIN	SQR	STEP	TAB	TAN
THEN	TO	VAL			
+	-	*	/	^	=
<	>	< =	> =	< >	

Im begrenzten Umfang ist DEF FN möglich.

11.2. Abweichungen bei den erlaubten BASIC-Befehlen

Hier werden nur eventuelle Abweichungen vom üblichen Gebrauch notiert.

AND

Es darf nur auf logische Ausdrücke (Vergleiche) bezogen werden. Die zugehörigen Ausdrücke sollten immer geklammert werden. Also z. B. (X > 45) AND (X < 70). Zuweisungen von logischen Ergebnissen an eine Variable sind zur besseren Übersicht und Verkürzung zulässig. Also z. B.:
Q = (A = 5) AND (B = 0): IF Q THEN ...

ASC

Nicht alle Computer liefern in allen Fällen exakt den gleichen Wert. Das ist jedoch für die Buchstaben weitgehend erfüllt. Die Nutzung der Funktion sollte daher mit Umsicht und für nur ein Zeichen erfolgen (Besonderheiten beim Commodore beachten!).

CHR$

Bei Werten, die kleiner als 32 sind (Steuerzeichen), empfiehlt sich Vorsicht. Die Besonderheiten des Commodore sind zu berücksichtigen.

DEF FN Name

Nur in der einfachen Form mit einer Variablen anwendbar.

DIM
Hat immer vor der Nutzung von Feldern zu erfolgen. In BASICODE sind maximal 2 Dimensionen, also z. B. DIM A(3,15), zugelassen. Das Feld beginnt stets mit dem nullten Element, also A(0,0).

FOR...TO...STEP..NEXT
Die Schleife wird mindestens einmal durchlaufen. STEP und der Wert danach dürfen weggelassen werden, dann beträgt die Schrittweite 1. Nach NEXT muß immer die zugehörige Variable stehen. Zu einem FOR ist auch nur ein NEXT zulässig. Das heißt, die Schleife kann nur an einer einzigen Stelle verlassen werden. Aus der Schleife darf nicht herausgesprungen werden. Vorzeitiges Verlassen über eine Bedingung erfolgt durch Setzen des Laufparameters auf seinen Endwert und Sprung zum NEXT.

GOSUB
Es muß eine Zahl und keine Variable als Zeilennummer folgen. Eine nicht existierende Zeilenzahl darf nicht verwendet werden. IF ... GOSUB ist nicht zulässig, richtig ist IF ... THEN GOSUB.

GOTO
Wie GOSUB, mit Ausnahme von 20 und 950. Auch wenn bei den meisten Rechnern IF ... THEN GOTO <Zeilennummer> bzw. IF...GOTO <Zeilennummer> funktioniert, soll konsequenterweise IF ... THEN <Zeilennummer> benutzt werden.

INPUT
Kann nur eine Variable übernehmen. INPUT A, B ist nicht zugelassen.

LOG
Bezieht sich auf die Basis e. Es ist zu beachten, daß in einigen Rechnern hierfür auch LN existiert, was nicht verwendet werden darf. Bei solchen Rechnern kann sich LOG fälschlicherweise auf den dekadischen Logarithmus beziehen. Die Bascoder sorgen für die richtige Umkodierung.

MID$
Erfordert drei oder zwei Werte. MID$(A$,5) ist erlaubt.

NEXT
Verlangt die Laufvariable, siehe FOR.

NOT
Verlangt Klammern, siehe AND.

ON
Nach ON darf die Variable nur die zulässigen Werte annehmen, also von 1 bis zur Anzahl der nach GOTO bzw. GOSUB stehenden Adressen.

OR
Siehe Bemerkungen unter AND.

PRINT
Zur Formatierung existieren nur das Komma, Semikolon und TAB. Mehrere Druckaufträge hinter PRINT müssen generell durch ein Semikolon getrennt werden. Es wird empfohlen, TAB oder Komma durch GOSUB 110 zu ersetzen.

REM
Es gibt nur REM und keine alternativen Zeichen, wie ! oder '. In einer REM-Zeile sollte kein Doppelpunkt verwendet werden.

RESTORE
Existiert nur ohne Zeilenzahl.

TAB
TAB(0) ist nicht erlaubt. Vorsicht! Es gibt einige Computer, die mit 1 zu zählen beginnen. Für die meisten ist die erste Position 0. Subroutine GOSUB 110 befreit von dieser Unsicherheit.

VAL
Nimmt bei nicht rein numerischen Argumenten in verschiedenen Rechnern unterschiedliche Werte an.

11.3. Kurze Übersicht der BASICODE-GOSUB-Routinen

Die rechts stehenden BASIC-Befehle zeigen Ähnlichkeiten auf und sollen den Überblick erleichtern. Sie wurden in erster Linie aus dem KC-BASIC gewählt.

20	Programmstart, System-Reset, Variable löschen usw.	CLEAR
100	Text-Modus einschalten und Bildschirm löschen	CLS
110	Cursor auf die Position HO, VE	LOCATE
120	Cursor-Position in HO, VE zurückholen	VGET, POS
150	Auffälliges Anzeigen von SR$; rechts und links 3 Spaces	kein
200	Daten einer eventuell gedrückten Taste in IN$ und IN	INKEY$
210	wie 200, jedoch mit Warten auf Tastendruck	kein
220	Holen des Zeichens aus Schirmposition HO, VE auf IN	VGET$
250	Erzeugen eines kurzen Aufmerksamkeitstones	BEEP
260	Zufallsvariable in RV mit 0 < = RV < 1	RND
270	Ausführen von garbage collection und Speicherplatz in FR	FRE(X)
280	Aus- bzw. Einschalten der STOP/BRK-Taste FR = 0 bzw. 1	kein
300	SR wird ohne Space in SR$ gewandelt	STR$
310	wie 300, jedoch als Zahl mit CT und CN formatiert	USING
330	Alle Kleinbuchstaben in SR$ in Großbuchstaben wandeln	kein
350	Übergabe von SR$ an den Drucker	PRINT#2
360	In neue Zeile mit Drucker (CRLF)	kein
400	Erzeugung eines Tons gemäß SV, SD und SP	SOUND
450	Warten von maximal SD*0.1 s auf einen Tasteneindruck	PAUSE
500	Eröffnen eines File mit Namen NF$ gemäß NF	OPEN
540	Aus dem File wird ein String an IN$ übergeben	LOAD*
560	SR$ wird in das File geschrieben	SAVE*
580	man schließe den Bestand mit Code NF ab	CLOSE
600	Graphischen Betrieb und Bildschirm löschen	SCREEN
620	Setzen eines Punktes in die Position HO, VE mit Farbe CN	PSET
630	Zeichnen einer Linie zum Punkt HO, VE in Farbe CN	LINE
650	SR$ an der Position HO, VE anzeigen (Grafik-Mode)	kein
950	Beenden des BASICODE-Mode	END

11.4. Nicht erlaubte BASIC-Befehle der KC-Rechner

!	AT	BEEP	BLOAD
BORDER	BYE	CALL*	CALL
CIRCLE	COLOR	CLEAR	CLOAD*
CLOSE	CLS	CONT	CSAVE*
CSRLIN	DEEK	DOKE	ELSE
END	FRE	INK	INKEY$
INP	INPUT#	INSTR	JOYST
KEY	KEYLIST	LINE	LINES
LIST#	LOAD#	LOAD*	LOCATE
NULL	OPEN	OUT	PAPER
PAUSE	PEEK	PI	POKE
POSE	PRESET	PRINT#	PSET
RND	RANDOMIZE	RENUMBER	RUN
SOUND	SPC	STOP	STRING$
STR$	SWITCH	TROFF	TRON
USR	VGET$	VPEEK	VPOKE
WAIT	WIDTH	WINDOW	(LN <-> LOG beachten)

11.5. Zeilen-Verwendung beim BASICODE

0 - 999
Bascoder (für jeden Computer anders).

1000
Erste Zeile des BASICODE-Programms. Sie muß folgende Form haben:

```
1ØØØ A = <Wert> : GOTO 2Ø: REM <Programmname>
```

1010 - 19999
Eigentliches BASICODE-Programm.

20000 - 24999
Subroutinen, welche unerlaubte Befehle verwenden (Tab. 11.4.). Sie sollen möglichst vermieden, zumindest aber in REM-Zeilen genau erklärt werden. Ein mögliches Beispiel sind farbige Bilder, welche in BASICODE-3 nicht möglich sind.

25000 - 29999
Eventuell benötigte DATA-Zeilen.

30000 - 31999
REM-Zeilen, die z. B. eine kurze Beschreibung des Programms und Literaturhinweise enthalten.

32000 - 32767
REM-Zeilen für den Namen und die Anschrift des Autors, Datum und andere formale Bemerkungen.

Der Zeilenabstand sollte im Programm möglichst in 10er Abständen gewählt werden. Dies bringt unter anderem Vorteile, wenn wirklich einmal Fehler bei der Übertragung oder Speicherung auftreten sollten. Die Zeilenlänge einschließlich der Zeilennummer und Space beträgt maximal 60 ASCII-Zeichen.

11.6. In BASICODE verbotene Variablen

1. Alle Variablen, die mit dem Buchstaben O beginnen.
2. AS, AT, DI, DI$, DS, D$$, EI, EI$, EL, ER, FN, GO, GR, IF, LN, SQ, SQ$, ST, TI, TI$ und TO.
3. PI, enthält aber nicht den Zahlenwert 3.14159.

11.7. Im Bascoder verwendete Variablen mit besonderer Bedeutung

A, CN, CT, FR, HG, HO, IN, IN$, NF, NF$, RV, SD, SP, SR, SR$, SV, VE und VG.

11.8. Kleines Fachwörterbuch

Es sind hier einfache Begriffe kurz erklärt. Eine Vollständigkeit wurde nicht angestrebt. Es soll vor allem dem Neuling eine leichtere Einführung in BASICODE (z. T. auch allgemein in die Rechentechnik) ermöglichen. Ausführlicher sind BASICODE-typische Begriffe häufig im Text erklärt und über das Sachwörterverzeichnis auffindbar.

Adresse Zahlenmäßiger Wert für einen Speicherplatz im Rechner.

Array Siehe Feld.

ASCII American Standard Code for Information Interchange; deutsch etwa: Amerikanischer Standard für den Austausch von Information. Es existieren gleichwertige Standards in allen Ländern. Dieser Begriff hat sich verallgemeinert und gilt als Kennzeichen für eine bestimmte Codierung von Daten, insbesondere von Buchstaben, Ziffern und Interpunktionszeichen.

Assembler Maschinennahe Programmsprache und zugehöriges Programm; hierbei werden Mnemoniks wie LD A,7 (lade Register A mit dem Wert 7) in für den Rechner direkt abarbeitbare Befehle 3E 07 automatisch übersetzt.

Asynchron Betrieb, Übertragung, bei der keine zeitliche Taktung erfolgt.

BASIC Beginners All purpose Symbolic Instruction Code; deutsch etwa: Symbolischer Code (Sprache) für Anfänger bei allen Anwendungen.

Bascoder Spezielles Programm, welches die Nutzung von BASICODE ermöglicht.

Baud Maßeinheit für die übertragenen Daten pro Sekunde, also gleichwertig mit bit/s.

Bediener-Oberfläche Art und Weise, wie ein Rechner bzw. ein Programm dem Anwender gegenübertritt.

Bestand Siehe File (hier meist durch die Übernahme aus dem Holländischen).

Betriebssystem Eine Zusammenfassung von meist kleinen Programmen, die jeder Rechner benötigt, um überhaupt auf niedrigstem Niveau betrieben werden zu können. Betriebssysteme haben oft spezielle Namen, wie z. B. CAOS beim KC85/3 oder CP/M bei vielen 8-Bit-Rechnern mit Diskette oder MS-DOS bei 16-Bit-Rechnern.

Bildpunkt Siehe Pixel.

Binär Codierung von Zahlen in der zweiwertigen Zahlenbasis. Binärwerte sind folglich durch eine Folge von 0/1-Werten gekennzeichnet.

Block Feste Struktur von Daten (z. B. Daten-Block bei der Übertragung) oder in Programmen: Blockstruktur als genaue Ablauffolge von Befehlen.

Bus Zusammenfassung von mehreren Leitungen zur Datenübertragung; z. B. Adreß-, Daten- oder Steuerbus. Nach der Anzahl der Datenleitungen unterscheidet man u. a. die 8- und 16-Bit-Technik von Rechnern.

Bit Grundeinheit in der Informationstheorie, in der Rechentechnik meist eine Speicherzelle, die den Wert 0 oder 1 annehmen kann.

Byte Zusammenfassung von 8 Bit, wird meist in HEX-Werten angegeben und kann die Werte 0 bis 255 dezimal oder 0 bis FF in HEX annehmen.

Check-Byte Etwa: Prüf-Byte zum Test auf Fehlerfreiheit.

Code Feste Zuordnung zwischen Zeichen, Objekten usw. zu einem Zahlenwert; siehe auch: ASCII, HEX und binär.

Compiler Programm, welches aus den Befehlen einer Programmiersprache maschinenlesbare direkt abarbeitbare Programme (z. B. .COM oder .EXE) erzeugt (siehe auch Interpreter).

Control [Etwas] steuern

Controler Steuergerät, z. B. Disketten- oder Bildschirm-Controler.

Control-Code Siehe Steuer-Code.

CPU Central Processing Unit; deutsch etwa: Zentraleinheit, Mikroprozessor, z. B. U880, Z-80, i8086.

CRC Cyclischer Redundanz-Code; eine Methode, Zusatzwerte zu berechnen, die eine sehr zuverlässige Prüfung auf Fehlerfreiheit ermöglichen.

Cursor Deutsch etwa: Schreibmarke ; Kennzeichen auf dem Bildschirm, das angibt, wo das nächste Zeichen erscheinen wird.

Datei Siehe File.

Debugging Fehler in Programmen beheben. Geht auf die alten Relaisrechner zurück. Dort hatte sich beim Aiken-Rechner eine Motte (engl. bug) in ein Relais gesetzt und wurde zerdrückt, wodurch das Relais und damit der Rechner nicht mehr arbeiten konnte.

Directory Inhaltsverzeichnis von Disketten und Festplatten.

Direkt-Mode Insbesondere bei BASIC und BASICODE unmittelbare Eingabe von BASIC- bzw. BASICODE-Befehlen oder Aufruf von Subroutinen.

Dummy Englisch Attrappe; meist ein Wert, eine Variable oder ähnliches, was zunächst nur formal oder vorläufig (ohne direkten Sinn oder Bedeutung) verwendet wird.

Dump Anzeige eines oder mehrerer Speicherinhalte.

Editor Programm zum Schreiben und Ändern von Texten und Programmen.

ENTER Von to enter; deutsch: eingeben. Vielfach Bezeichnung für eine Taste, mit der eine Eingabe abgeschlossen wird. Andere Bezeichnung: CR von carrige return (Wagenrücklauf), dann Steuer-Code mit dem Wert 0D HEX.

Feld Eine Zusammenfassung von Variablen zu einem Ganzen, muß mit DIM eingeleitet werden, z. B. DIM A$(7); englisch: Array.

Festplatte Speicher ähnlich einer Diskette, jedoch mit wesentlich höherer Speicherkapazität und viel schnellerem Zugriff; besondere Bezeichnung ist Winchester-Speicher.

File Zusammenfassung von Daten, Bildern, allgemein Information zu einem Ganzen, meist maschinenlesbar. Etwa gleichwertig ist Datei oder Bestand.

Formatieren Etwas in einer bestimmten Gestalt, Form darstellen. Bei Disketten: Sie durch eine bestimmte Datenstruktur erst für die Anwendung brauchbar machen.

Funktion Allgemein die Aufgabe, welche ein System (eine Struktur) zu leisten hat.

garbage collection Eigentlich Müll- (Abfall-) Beseitigung, spezielles Verfahren zur Speicherverwaltung bei String.

Grafik-Karte Hardware für die Bildschirmdarstellung (meist in Pixel-Grafik) mit unterschiedlicher Pixel-Anzahl. Bei den IBM-kompatiblen Rechnern existieren mehrere genormten Grafik-Karten, z. B. MDA, CGA, EGA, VGA und Hercules.

Grafik-Mode Betriebsweise (u. a. in BASICODE), welche die Darstellung von Bildern erlaubt. In der Regel als Pixelgrafik realisiert.

Hardware Insbesondere bei Rechnern, alles, was stofflich existiert, z. B. Schaltkreise, Leiterplatten, Gehäuse, Tastatur, Bildschirm, Kassettenrecorder, Diskettenlaufwerke, Drucker usw. Das Gegenteil ist Software.

Hardcopy Ausdruck des Bildschirminhaltes auf den Drucker.

HEX Von hexadezimal; Darstellung einer Zahl im 16er-Zahlensystem. Es werden dabei die zusätzlichen Ziffern-Zeichen A = 10; B = 11; C = 12; D = 13; E = 14 und F = 15 verwendet. Die Zahl A7 bedeutet dann z. B. (A = 10)*16 + 7 = 167.

hidden-line Deutsch: verdeckte Linie; linienhafte Grafik-Darstellung von drei-dimensionalen Bildern, bei der die unsichtbaren (durch räumlich davorliegende Teile verdeckten) Linienteile nicht dargestellt werden.

Integer In der Rechentechnik und Mathematik gleichwertig mit ganzzahlig.

Interface Deutsch: Schnittstelle; verbindet zwei Systeme, z. B. einen Rechner mit dem Drucker (Drucker-Schnittstelle). Schnittstellen sind meist genormt. Es gibt dann genau festgelegte Parameter.

Interpreter Programm einer Sprache, welches die Befehle unmittelbar in für den Rechner abarbeitbare Befehle übersetzt. Während ein Compiler das vollständige Programm als Einheit (meist in mehreren aufeinanderfolgenden Schritten) übersetzt, arbeitet ein Interpreter das Programm in kleinen Schritten interaktiv ab.

	Ein Compiler kann mit einem Dolmetscher verglichen werden, der nur den ganzen Text schriftlich übersetzt zurückgibt. Ein Interpreter arbeitet dagegen wie ein Simultan-Dolmetscher.
KC-BASIC	BASIC-Dialekt, der bei den KC-Rechnern verwendet wird.
KC-Rechner	Kurzbezeichnung für die folgenden Rechner: a) vom Kombinat Robotron: KC 85/1 und KC 87 b) vom Kombinat Mikroelektronik: KC 85/2; KC 85/3 und KC 85/4.
Keyboard	Die Tastatur.
Kompatibiltät	Austauschbarkeit, z. B. bezüglich Hardware von Disketten oder Kassetten, bezüglich Software von Programmen. Hier liegt die hohe Leistungsfähigkeit von BASICODE.
Laufvariable	Variable, die sich in der FOR-NEXT-Schleife schrittweise ändert. In FOR X = 1 TO 100 ist X die Laufvariable. Sie muß bei BASICODE nach NEXT gemäß NEXT X wiederholt werden.
LF	Line feed, deutsch: Zeilenvorschub; ein Steuer-Code mit dem Wert 0A HEX bzw. 10 dezimal.
Logik	Hier im engeren Sinne die Methode der Entscheidung zwischen zwei Möglichkeiten: wahr und falsch.
Logische Variable	Variable, welche nur die Wahrheitswerte wahr und falsch annehmen kann und dies über zwei Zahlenwerte, meist -1 und 0, kennzeichnet.
Maschinencode	Maschinensprache (siehe Assembler). Direkt abarbeitbare Befehlsfolge, meist als HEX-Code gegeben.
Memory	Der Speicher.
Mnemonik	Leicht zu merkender direkter Befehl (siehe Assembler).
Menu, Menü	Sichtbar angezeigte Möglichkeit der Auswahl zwischen verschiedenen Programmteilen (vergleiche Bedieneroberfläche).
Mode	Betriebszustand eines Systems; bei Rechnern meist als Auswahl aus mehreren möglichen Zuständen.
Modul	Abgeschlossene Einheit, sowohl als Hard- wie als Software.
Monitor	a) Hardware: Etwa gleichwertig mit Bildschirmgerät, z. T. unter Einschluß der Tastatur. Dann ähnlich wie Terminal. b) Software: Einfaches Betriebssystem.
Option	Etwa: zusätzliche Möglichkeit, frei wählbares Zubehör.
Optimieren	Eine Methode, bei der zwischen widersprüchlichen Lösungen zum gemeinsamen Gesamtvorteil ausgewählt wird.
Parameter	Wählbarer Wert.
Pixel	Einzelner Bildpunkt; in der Regel mit zugeordnetem Speicherplatz.
Pixel-Grafik	Grafik, welche unmittelbar auf die einzelnen Pixel eines Bildschirmes zugreift. Das Gegenteil ist Vektorgrafik.
Pointer	Zeiger; in BASICODE u. a. über Felder realisiert.

Programm Zusammenfassung von Befehlen, welche eine Aufgabe lösen. Ein Programm kann in unterschiedlichen Sprachen geschrieben werden: Assembler, BASIC, BASICODE, PASCAL usw.

Programmierung (strukturierte) Methode bei der Programmierung, um die möglichen Programmierfehler wesentlich zu reduzieren. Typische Kennzeichen sind: Es darf jeder Programmblock nur einen einzigen Ein- und Ausgang besitzen, es sind (nicht in BASICODE möglich) lokale und globale Variablen zu unterscheiden und es müssen zwischen Programmodulen (z. B. Subroutinen) eindeutige Übergabeparameter festgelegt sein.

Pseudografik Eine spezielle Variante der Pixel-Grafik mit erheblich geringerer Leistungsfähigkeit. Bei ihr kann nur sehr indirekt zu den einzelnen Pixel zugegriffen werden. Es sind Grafik-Zeichen definiert, die ähnlich wie ASCII-Zeichen verwendet werden.

RAM Random Access Memory; deutsch: Speicher mit beliebigen Zugriff, meist über die Adresse. Der Inhalt kann beliebig gelesen und neu geschrieben werden (vergleiche auch ROM).

Raster-Grafik Siehe Pixel-Grafik.

Redundanz Deutsch: Weitschweifigkeit; Ergänzung oder vorhandener Anteil in einer Information, der eigentlich überflüssig ist. Die Redundanz wird oft dazu verwendet, um Fehler zu erkennen oder zu beheben.

Rekursion Prinzip des sich selbst Aufrufens einer Funktion oder Routine; sehr leistungsfähige Methode der Rechentechnik. BASICODE enthält leider keine direkte Möglichkeit zur Rekursion. Hier muß sie indirekt vom Programmierer verwirklicht werden.

ROM Read Only Memory; deutsch: nur lesbarer Speicher. Seine Information wird ihm bei der Herstellung unveränderlich eingeprägt (vergleiche RAM).

RS 232 Siehe V24.

Rundung Mathematische Methode zur ungefähren Darstellung von gebrochenen Zahlen.

Schalter Üblich Hardware, aber auch Software; beim Aufruf eines Programms oder innerhalb eines Programms existieren oft mehrere Möglichkeiten. Die Auswahl zwischen ihnen erfolgt durch einen Schalter, z. B.: SAVE "Name",A. Hier ist das Komma der Schalter und A die Auswahl. Läßt man ",A" weg, so wird das Programm in der Token-Form gespeichert, mit dem Schalter ",A" dagegen im ASCII-Format.

Schmitt-Trigger Elektronische Schaltung mit Hysterese-Verhalten, wichtig zur Störunterdrükkung.

Schnittstelle Siehe Interface.

Screendump Siehe Hardcopy und Dump.

Seiteneffekt Indirekte, meist nicht beabsichtigte Änderung des Wertes einer Variablen. Tritt vor allem bei globalen Variablen störend auf und war ein Anlaß zur Entwicklung der strukturierten Programmierung.

Software Zusammenfassung von Programmen, Daten usw. Die Software ist notwendig, um einen Rechner (Hardware) betreiben zu können. Sie bestimmt auch wesentlich dessen Leistungsfähigkeit.

Steuer-Code Auch: Steuer-Zeichen; Bytes mit einem Zahlenwert unter 32 dezimal bzw. 20 HEX, welche bestimmte Wirkungen hervorrufen, z. B. CR = Carrige Return = Wagenrücklauf = 0D; aber meist Abschluß einer Eingabe. Sie sind weitgehend im ASCII-Code festgelegt.

String Zeichenkette; Zusammenfassung von ASCII-Zeichen zu einer Einheit.

Struktur Eine feste Anordnung von Elementen und deren Verbindung. Sie bewirkt meist eine oder mehrere Funktionen.

Terminal Eigentlich Datenendstation, oft aber allgemeiner oder auch im Sinne von Tastatur mit Bildschirm benutzt.

Token Englisch etwa: Festgelegtes; bei Interpretern existieren die reservierten Wörter in zwei Formen, z. B. als ASCII-Kette PRINT oder als Token 9E. Die Token in BASIC und BASICODE sind also Kurzformen (HEX-Code) für die Befehle.

Variable Reservierter Speicherplatz, der über den Namen der Variablen festgelegt wird. Es können u. a. lokale und globale Variablen unterschieden werden. Globale Variablen sind für ein ganzes Programm gültig, lokale nur innerhalb eines Unterprogramms. In BASICODE existieren nur globale Variablen.

Vektorgrafik Darstellung von Bildern usw. mittels Vektoren, d. h. gerichteten Linien, wobei nur die Koordinaten der Endpunkte im Speicher existieren; meist das Gegenteil von Pixel-Grafik.

Viren Eigentlich in der Mikrobiologie: Nicht selbständige Lebewesen, die sich auf Kosten anderer vermehren und diese dabei zerstören. Es gibt sich ganz ähnlich verhaltende Programme bzw. Programmteile, die sich auf Kosten nützlicher Programme vervielfachen und dabei die eigentlichen Programme zerstören. Sie können sich auf Systemen mit Disketten und/oder Festplatten, jedoch insbesondere im Rechnerverbund seuchenähnlich zerstörerisch ausbreiten.

V24 Genormte Schnittstelle zur seriellen Übertragung mit einstellbarer Datenrate (in Baud), fast gleich mit RS 232.

Winchester-Speicher Siehe Festplatte.

Wort, reserviertes Zeichenfolge in einer Programmiersprache mit festgelegter Bedeutung; in BASICODE z. B. FOR, NEXT, INPUT usw.

11.9. Daten verschiedener Rechnertypen

	KC87	KC85/3	KC85/4	C 64	C plus 4	Z1013	AC1	ATARI	Spectrum	CPC
Taktfrequenz (MHz)	2.46	1.75	1.75	0.985	1	2/4	2.0	1.773	3,5	4,0 (3,2)
BASIC-ROM (kByte)	8	8	8		16			abgesch.	12	16
Betriebssystem (kByte)	8	8	12	8	16	5	2 ... 4	16/8	16	16
freier RAM min. (kByte)								21	26	31
freier RAM max. (kByte)			33	32	40	40		48	64/128	
Bildschirmzeilen	24	32	32	25	25	32	32	24	24	24 + 1
Zeichen / Zeile	40	40	40	40	40	32	64	40	42	80/40/20
Grafikzeilen	48	240	240	200	200	48	64	192	192	400
Grafikpunkte pro Zeile	64	320	320	288	250	64	128	256 (320)	256	640
SP tiefster Ton	2	0	0	0	45	0	0	8	0	11
SP höchster Ton	27	127	127	06	127	119	127	127	127	112
Rechengenauigkeit	7	7	7	7	9	7	7	9	7	9
kleine Buchstaben	ja	ja	ja	ja	ja	ja	ja	ja	ja	ja

Literaturverzeichnis

[1] Haubrich, J. u.a: Het BASICODE-3-boek. 2. Druck; Kluwer Technische Boeken B.V. Deventer-Antwerpen, 1988

[2] Grote, U.; Völz, H.: BASIC Einmaleins des Programmierens. Urania-Extra, Urania-Verlag, Leipzig 1988

[3] Sellhorn, W.; Branstner, G.: Das Tier lacht nicht. 4. Aufl. Eulenspiegel- Verlag, Berlin 1974 (S. 72 ff.)

[4] Mozart, W. A.: KV. 294d. Musikalisches Würfelspiel. Eine Anleitung "Walzer oder Schleifer mit 2 Würfeln zu komponieren, ohne musikalisch zu sein, noch vom Komponieren etwas zu verstehen". Verlag Edition B. Schott's Söhne, Mainz 1956. Nr. 4474

[5] Völz, H.: Computer und Kunst. akzent- Reihe Heft 87. Urania-Verlag, Leipzig, Jena, Berlin 1988

[6] Völz, H.: Graphic auf dem KC 85/2. radio fernsehen elektronik. 35 (1986) 1, 21-23

[7] Völz, H.: In BASIC effektiv programmieren - Auch mit Kleinstrechnern. Die Wirtschaft, Berlin 1989

[8] Groh, J.: Kleincomputer-Fibel. Akademie-Verlag, Berlin 1986

[9] Fischer, P.: BASIC für Anfänger. Verlag Die Wirtschaft, Berlin 1987

[10] Werner, D.: BASIC für Mikrorechner - Programmentwicklung, Sprachelemente, Anwendungen. VEB Verlag Technik, Berlin 1987

[11] Wirth, N.: Algorithmen und Datenstrukturen. B. G. Teubner GmbH, Stuttgart 1975

[12] Knuth, D. E.: The Art of Computer Programming. Addison-Wesley, Reading. Vol. 3 / Sorting and Searching 1975

[13] Völz, H.: Grundlagen der Information. Akademie-Verlag, Berlin (voraussichtlich 1990)

[14] Völz, H.: Kleines Lexikon der Speichertechnik. Reihe Automatisierungstechnik, Band 224. Verlag Technik, Berlin 1987

[15] Die Routine zur Berechnung der Linien wurde dem Programm "CAOS-NEU" entnommen.

[16] Farrour, J.: Microsoft BASIC Decoded & Other Mysteries. IJG Computer Services 1981

[17] Gemini Microcomputer Vertriebs-GmbH. NASCOM 8K BASIC Programmierhandbuch N.A.S. Elektronische Halbleiter, GmbH München 1980 und Interpreter

[18] Tandy Corporation 1980: TRS-80 Model III, Operation and BASICLanguage Reference Manual

[19] VEB Mikroelektronik "Wilhelm Pieck" Mühlhausen: KC 85 BASIC Interpreter und Beschreibung

[20] Heyder,F., u. a.: Der Amateurcomputer AC 1; Artikelserie in der Zeitschrift FUNKAMATEUR 1983, 1984, 1985, 1986, 1989

[21] Püschel, H.: "Wirksamer Rechtsschutz für schöpferische Leistungen auf dem Gebiet der Computersoftware", in: Staat und Recht, Staatsverlag der DDR, 1989, Seite 228 ff.

[22] Herd, B.; Stiller, A.: Supertape für Schneider CPC464, c't, (1985) H. 9; S. 78- 83

[23] Brückmann, Englisch, Gerits, Steigers: CPC 664/6128 intern, Data Becker, 1985

Sachwörterverzeichnis

Analog-Kanal 50
AND 193
Anzeigen, auffällig 19
Arbeit mit Feldern 146
Arithmetik-Grenze 23
Array 34
ASC 193
ASCII
-File 15, 49, 54, 63, 67, 75
-Programm 15
asynchron 53
ATLAS-Programm 152
Audiokassetten 191
Aussteuerung 50
Autorenschaft 182

back-up 57, 191
Band
-geschwindigkeit 60
-material 59
Bascoder 11, 14, 31, 61, 63, 65, 67, 69, 71
BASCON 100
BASIC
- für Fortgeschrittene 177
-Befehle 9, 14, 193
-Befehle, verbotene 196
-Direkt-Mode 14
-Lehrgang, Programmieren 176
-Programm 63
-Sendungen, DDR 175
BASICODE
-, Grenzen 11
-, Nachteile 11
-, Sprachteil 13
-, Stiftung 9
-, System 12
-, Vorteile 35
-Fehler 44
-Geschichte 173
-Idee 174
-Mode 14
-Programm 14, 49
-Programme 54
-Protokoll 174
BEEP 195
Beschreibung, Bascoder 61
Bestände 68
Betriebssystem 14
Bildschirm
-format 11, 18
-Koordinaten 24

Binär-File 49
Bit-Niveau 50
Block
-nummer 56
-struktur 38
BUBBLE-Programm 22
Büroklammer 58
Byte-Niveau 53

CASE-Struktur 41
Centronics-Schnittstelle 136
Check-Byte 55
CHR$ 193
CLEAR 16, 195
CLOAD* 29
CLOSE 29, 195
CLS 195
Codierung 50
Commodore 19, 21, 22, 32, 92, 167
Compiler 12
Computerclub 177
Computermagazin 178
CSAVE* 29
Cursor
-Befehle 19
-Grafik 25

Dachabfall 52
Darstellung
-, interne 15
-, perspektivische 161
3-D-Darstellung 162
DATA-Zeilen 44, 151
Datasette 128
Datei-Arbeit 151
Daten
- von Rechnern 203
-, retten 151
-austausch 11
-File 29, 55
-format 11
-format, internes 49
-Programm 18
-satz-Globus 154
-träger 191
Decodierung 53
DEF FN 193
DELETE 19
Detektor-Schaltung 50
DIM 194
Disketten 58

Dormeier, F. 178
Drucker
-, Anschluß 70, 82
-Ausgabe 32
Durchläufe 42

Eigentum, geistiges 181
Einstellungen, Recorder 59
END 16, 195
Ende-Byte 56
ENTER 19
Entscheidung 38
-, mehrstufig 150
Erdachse 158
Erlernbarkeit 13

Farbfernseher 58
Farbwert 25
Fehler
-, Arten 44
-behandlung 13
-freiheit 55, 57
-suche 44
Feld 34
File
-Name 29
-typ 49
FILE-Programm 30
FKTD-Programm 162
FM 51
FOR 194
FORMAT 32
-Programm 23
FRE 195
Fremdaufnahmen 59
Frequenzbereich 49, 50
FRQ 72
Funktions-Definition 162

garbage collection 23, 45
Gedicht 147
Geschwindigkeit 46, 47
Gleichstromfrei 50
Globus 158
GOSUB 194
-, Anwendungen 41
-Routinen 13, 195
GOTO 37, 194
-Befehl 38
Grafik
-Cursor 25
-Mode 18, 23, 25, 27
-Schirm 24
GRAFIK-Programm 25

Haftmagnete 58
HC901 71
Hiddenline-Darstellung 161
HiFi-Gerät 52
Horizont 162
Hydra-Problem 16

IEC-Klassen 58
INKEY$ 195
INPUT 194
Interaktivität 13
Invertierung, Bit7 55

Kanal
-, analoger 50
-Abschnitte 50
-Eigenschaft 50
Kassette 11, 49
Kassetten
-, Umgang mit 56, 57
-Interface 31
-recorder 52, 59
-typen 58
Kompatibilität 11
Kontroll-Byte 55
Kopf, Justage 59
Kurzkassetten 58

Landkarte, Projektion 153
Laufparameter 43
Leerzeichen 16, 48, 141
Lesbarkeit 36, 46, 48
Lesen 29
LIED-Programm 28
LINE 195
Linien, verdeckte 161
LIST 31
LIST#1 31
LOAD#1 31
LOAD* 195
LOCATE 195
LOG 194
Logik 34
Logik-Befehle 26
LPRINT 33

Magnetfelder 58
Maschine, virtuelle 15
Maschinencode 178
McCarthy 17
Menü-Aufbau 148
MFM 51
MID$ 194
Minimal-BASIC 13
Modul 37
MODULA2 37
MOPS-Programm 145
MOZART-Programm 166
Musik 11, 33

N-ECK-Programm 26
Newline 33
NEXT 194
NEXT X 43
Nichtlinearitäten 51
Nonsens-Gedicht 145
NOT 194
Noten codieren 171
Noten einlesen 167
Nulldurchgang 50, 52

ON 194
OPEN 29, 195
Optimierung 46
OR 194

PASCAL 37
PAUSE 195
Phasen
-fehler 50, 52
-gang 50
Pixel 11
-zahlen 32
Plastehülle 57
POEME-Programm 147
Pointer, sortieren 47
POS 195
PRINT 194
PRINT USING 23
PRINT#2 33
Programm
-, lesen 36
-, Struktur 54
-ATLAS 152
-Aufbau 143
-Autor 182
-BUBBLE 22
-DATEN 18
-dokumentation 48
-Entwicklung 143
-FILE 30
-FKTD 162
-FORMAT 23
-GLOBUS 158
-GRAFIK 25
-HYDRA 16
-LIED 28
-Mode 14
-MOPS 145
-MOZART 166
-N-ECK 26
-POEME 147
-ROMZA 141
-STOP 20
-struktur 38, 39, 41, 43
-TASTE 20
-TEXTUM 21
-ZUTEXT 19
Programme, weiterentwickeln 182
Programmieren 63
Programmierkompetenz 35
Programmierung, strukturierte 35
Prosa-Gedichte 147
PSET 195

Qualitätskassetten 58

RANDOMIZE 18
Rauschabstand 52
Rechtecksignal 52
Redundanz 53
REM 178, 195
REM-Zeilen-Nutzung 144
RENUMBER 64
RESTORE 195
Römische Zahlen 141
ROMZA-Programm 141
Rückspulen 57
RUN 16
Rundfunk 11, 49

SAVE* 195
Schallplatte 10, 191
Schallplatten-Kurs 177
Schleifenstruktur 42
Schmitt-Trigger 50
Schreiben 29
Schulfunk 175
Schwächen 36
SCREEN 195
Sicherheitskopie 57
Signale-H/L 50
Software
-recht 181
-Service 177
Sound 28, 33, 195
Spaghetti-Code 40
Spaltenzahlen 32
Speicherplatz 46
-Minimierung 47
Spektralkomponente 51
Sprachkompetenz 35
Start
-Bit 53
-Byte 55
STEP 194
Stereo 59
-recorder 52
Steuerzeichen 19
STOP 16
STOP-Programm 20
Stopp-Bit 31, 53
Störabstand 50
STR$ 195
String
-, reservierter 19
-raum 46
Subroutinen 143
Systemabsturz 69

TAB 195
Taste unwirksam machen 20
TASTE-Programm 20
Temperatur 57
Test
-programm 12, 36
-programm KC85/3 72
Text-Mode 18
TEXTUM-Programm 21
Token-File 15
Tonausgabe 28, 33
Tricks 37
TTL 50

Übersetzen von BASICODE 15
Übertragungs
-fehler 15
-rate 51
Ungeduldige 14
Urania-Extra 177
Urheber
-recht 181
-schutz 181
USING 195

VAL 195
Variable 13, 33, 37
Variablen
- für Bascoder 197
-, logische 26
-, verbotene 197
VERIFY 57
Vervielfältigung 181
Verzweigung 39
VGET 195
Vielfachbefehl 47
Viren 60
Vorton 55

WHILE-WEND 42
Wortschatz 148
Würfelspiel, musikalisches 166

Zahlen, römische 141
Zeilen
-länge 16
-Numerierung 17
-Verwendung 196
-zahl 31
Zeiten, arithmetische Operationen 46
Zivilrecht 181
Zufalls
-melodie 170
-Variable 18
ZUTEXT-Programm 19
Zuverlässigkeit 57

BASICODE erfüllt die Wünsche vieler Computer-Freunde: Den direkten Austausch von Daten und Programmen zwischen verschiedenen Rechnern. Hierzu wurden ein einheitliches Datenformat und eine BASIC-ähnliche Sprache entwickelt. So ist es möglich, daß Texte, Grafik und Tonausgabe universell nutzbar sind.
BASICODE ist also das ideale Kommunikationssystem für Computerclubs, Arbeitsgemeinschaften, Schulen usw. Aber auch der „Stille Programmierer" wird viel Interessantes an BASICODE entdecken. Um die hohe Universalität erreichen zu können, benötigt jeder Rechnertyp ein spezielles Zusatzprogramm, den Bascoder. Für die wichtigsten Rechner befinden sich die Bascoder auf der zu diesem Buch gehörenden Schallplatte.